CABS Viewer Engine ™

Carrier Access Billing System
BOS BDT File Format
and Reporting Utility

User's Manual

Version 1.8

CABS Viewer Engine tm

CABS BOS BDT File Format and Reporting Utility User's Manual

Version 1.8

Lymeware Corporation
Post Office Box 1027
Old Lyme, Connecticut USA

2008-18-CVEUM-0018

2008-18-CVEUM-0018

Contents

Figures

Tables

Welcome

Thank you for using the CABS Viewer Engine, created by Lymeware Corporation. The CABS Viewer Engine product is designed to allow you to embed our powerful CABS BOS record-processing engine into your own internal billing process or products.

Most importantly, our CABS Viewer product line allows CLECs and other service providers to seamlessly blend their CABS billing data in to whatever OSS or billing system currently in use. The instructions in this guide will introduce you to integration concepts and help you get familiar with the fundamentals of using your CABS Viewer Engine product.

About this Guide

This guide is current with the details of operation for Lymeware's CABS Viewer Engine, version 1.8. It is designed for developers and software architects who are responsible for adding the CABS Viewer Engine to their internal billing processing or product.

The information in this guide describes how to link the CABS Viewer Engine (CVE) Dynamic Linked Library (DLL) or Shared Object Library (SO) to a current application with a simple Application Programming Interface (API).

Readers are expected to have programming or software development knowledge, and should be generally familiar with:

- A target programming language, such as ANSI C, C++, or Visual C++,
- The use of the target compiler and linker,
- Optionally, the use of a development graphical integrated development environment (IDE), such as Microsoft's Visual IDE or the Eclipse IDE

For more information on the CABS Viewer Engine capabilities and details on output options please refer to the latest version of *The CABS Viewer User Guide.*

A Note on Versioning

CABS Viewer Engine uses the version nomenclature of M.N, where M is the major release version and N is the minor release number. This arrangement matches the leading part of a CABS Viewer version of M.N.B.R. CVE releases and version numbers are aligned with CABS Viewer commercial product releases and versions (and support the same BOS version as the matching CABS Viewer). For example, CABS Viewer Engine 1.5 supports BOS 44, just like CABS Viewer 1.5.44.0 does.

Introducing CABS Viewer

The CABS Viewer product line (which includes both the CABS Viewer and the CABS Viewer Engine products) enables organizations to efficiently format, organize, and display information on all aspects of their carrier access billing data. This same billing data, once in a usable form, can now be inserted in to databases, used in spreadsheets to generate reports, or imported into existing accounting or billing systems.

The major benefits of the CABS Viewer products are:

Choice of Platform Support:

- Supports the platforms you use; including Microsoft Windows, Sun Solaris, and several Linux distributions (including Red Hat Enterprise Linux)
- Operation is exactly the same, regardless of platform
- Simple to use and easy to operate

Enabling Downstream Billing Automation:

- CABS Viewer output can be easily shared and processed
- Filter and format CABS data to your billing system's exact input requirements
- Special CSV-database format for import into most common enterprise database management systems, including Oracle, DB2, Informix, MySQL, Sybase, and MS SQL Server.
- Valid well-formed XML can support many standard Telecom billing systems

Useful for Revenue Assurance:

- Create user-friendly output, easily manipulated with Microsoft Excel, OpenOffice Calc or several other spreadsheet products.
- CSV or XML data can support many standard Telecom billing systems

Display Custom Billing Reports:

- CABS Viewer output is very easy to manipulate using may script languages, including Perl, Python, and Ruby – all of which can natively read CSV and XML data for further downstream processing or reporting.
- Lymeware is also available to deliver customer-specified custom reports as an additional service.

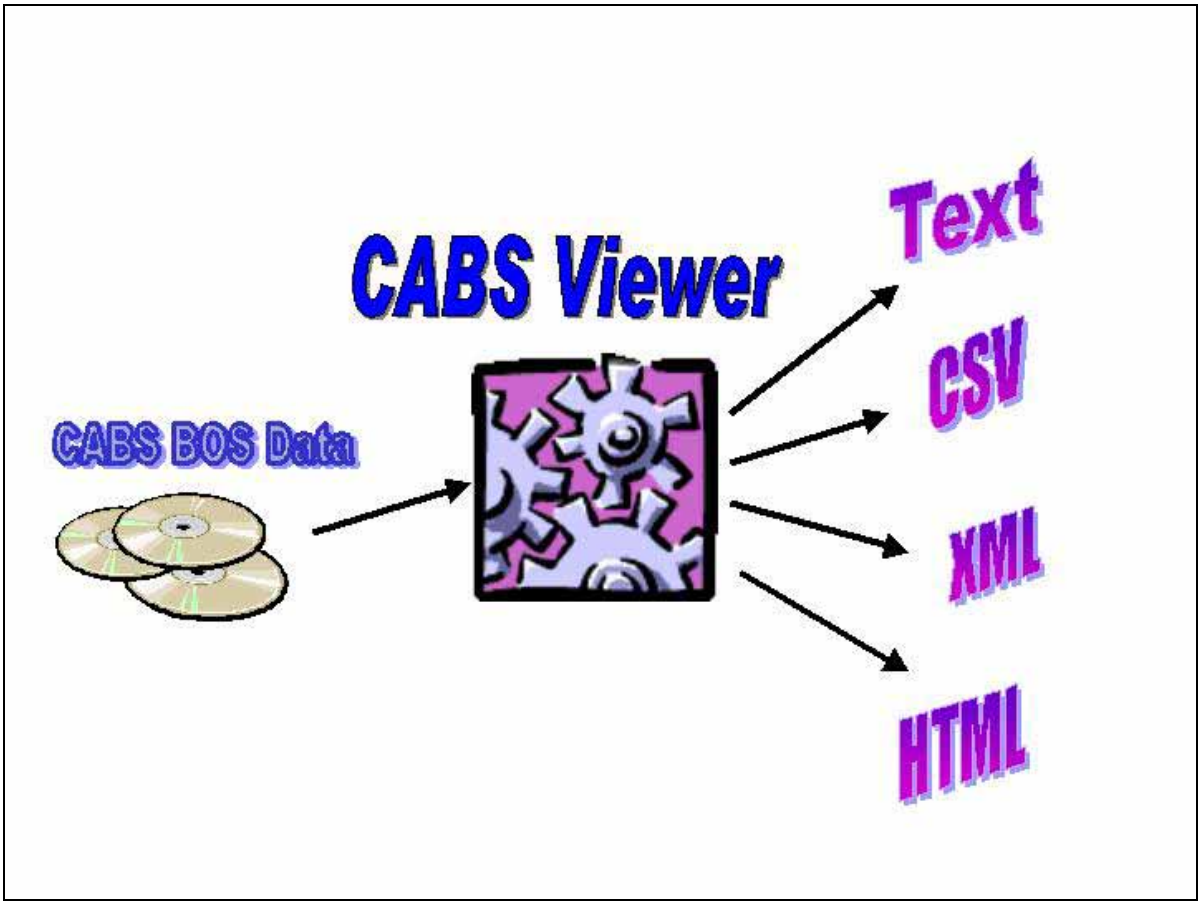

Figure 1. CABS Viewer in Action

CABS Viewer products are built on established server technologies and widely support data and industry specific standards, including:

- **CABS BOS BDT**, or Carrier Access Billing System Billing Output Specifications Billing Data Tape record formats and data presentment formats, the industry standard billing record transport format.

- **CSV**, or Comma Separated Value or comma-delimited format, for use with most spreadsheet products and a common database import format,

- **XML**, or Extensible Markup Language, for input to other billing or accounting systems or for further XSLT and XML processing, and

- **HTML**, Hypertext Markup Language, for display either locally or on a web server with any standard web browser

The CABS Viewer Engine is supported on the Linux, Solaris and Windows operating systems.

Getting Started with CABS Viewer

Before You Start

This guide assumes that the resources you need to access the system are available and that you are familiar with how to use them. If you are not sure whether your system meets the requirements or how to use required third-party tools (primarily a web browser), talk to your manager or system administrator.

Technical Requirements

Before you begin using the system, ensure that you have the appropriate software installed and configured on your system. All you will need is –

- **One of the following platforms running one of these operating systems:**
  - Red Hat Enterprise Linux 3 or 4 or 5 on Intel IA32/x86 platforms. Other Linux distributions may also be supported
  - Sun Microsystems Solaris 9 & 10 on UltraSPARC platforms
  - Microsoft Windows 2000 & XP on Intel IA32/x86 platforms. Note: Vista has not been tested at this time.

- **A supported development environment on your target platform.**

  CABS Viewer Engine has been tested with and supports a variety of development environments. The following environments are known to work with it:
  - GCC & G++ on the Linux target environment (as a shared library)
  - GCC & G++ on the Solaris target environment (as a shared library)
  - Microsoft Visual C++ 6.0 and Visual C++ 2005 and higher - www.microsoft.com

Installing CABS Viewer Engine

The CABS Viewer Engine product is available for

- Linux on ia32/Intel x86 (reference platform is Red Hat Enterprise Linux 4 x86 version),
- Sun Microsystems Solaris on UltraSPARC (reference platform is Solaris 10 SPARC version), and
- Microsoft Windows on ia32/Intel x86 (reference platform is Windows 2000 Service Pack 4 x86 version)

A typical CABS Viewer Engine installation consists of three actions:

1. Download the correct install package from the Lymeware website or request the correct package from Lymeware Sales,
2. Perform the platform specific package installation tasks (as detailed below),
3. Request and install the license key

The pre-installation checklist consists of:

- ❑ Print out this manual
- ❑ Install the development environment on the target machine
- ❑ Be sure the target machine has Internet access or download the install package on another Internet-enabled machine
- ❑ Complete the License Request form (see Appendix A)

Additionally, proceed to the correct platform installation section below.

Linux Package Installation

Linux users should download the RPM version of the CABS Viewer Engine installation file, via FTP or HTTP. The specific URL to use will be provided by the Lymeware Sales staff.

This file can be installed with the RPM command (logged in as root):

```
rpm -i <rpmfile>
```

This will install the CABS Viewer Engine files to `/opt/cve/src`.

Sun Solaris Package Installation

Solaris users should download the PKG version of the CABS Viewer Engine installation file, via FTP or HTTP. The specific URL to use will be provided by the Lymeware Sales staff.

This file can be installed with GZIP and TAR commands (logged in as root):

```
gzip -dc <tar.gz_file> | tar xvf -
```

This will install the CABS Viewer Engine files to **`/opt/cve/src`**.

Microsoft Windows Package Installation

Windows users should download the ZIP file version of the CABS Viewer Engine installation file, via FTP or HTTP. The specific URL to use will be provided by the Lymeware Sales staff.

This file can be unzipped with WINZIP or 7-UP and should be unpacked with the "Use Folder Names" option.

Requesting a license key

All new users need to request a license key to operate CABS Viewer Engine on any platform.

A specific license key file will be required to run the CABS Viewer Engine. Lymeware will supply this license file if the following information is supplied via email to Lymeware Sales (see Appendix A for the License Request form):

- Customer/Company Name:
- Product Name: **CABS Viewer Engine**
- Platform: [either **Linux**, **Solaris**, or **Windows 2000/XP**]
- Target Machine IP Address:
- Target Machine Host ID: (only needed for Solaris machines)
- Contact Person:
- Contact Phone Number:
- Contact E-Mail Address:

A digital license request form is also available for completion and submission to Lymeware. The license file will be delivered to the Contact E-Mail Address. In all cases, regardless of platform, the license key should be renamed to **license.dat** and copied to the same location as the CABS Viewer Engine library (**libcvengine.so** for Linux or Solaris or **cvengine.dll** for Windows).

A 30-day fully functional evaluation license is available from Lymeware's sales department. Lymeware Sales may be contacted at sales@lymeware.com.

Using CABS Viewer Engine

The CABS Viewer Engine is very powerful but provides a very simple interface to allow your external billing system or product to use the power of CABS Viewer Engine in your own solution.

Regardless of your target platform and the form of your CABS Viewer Engine, the basic operations are the same.

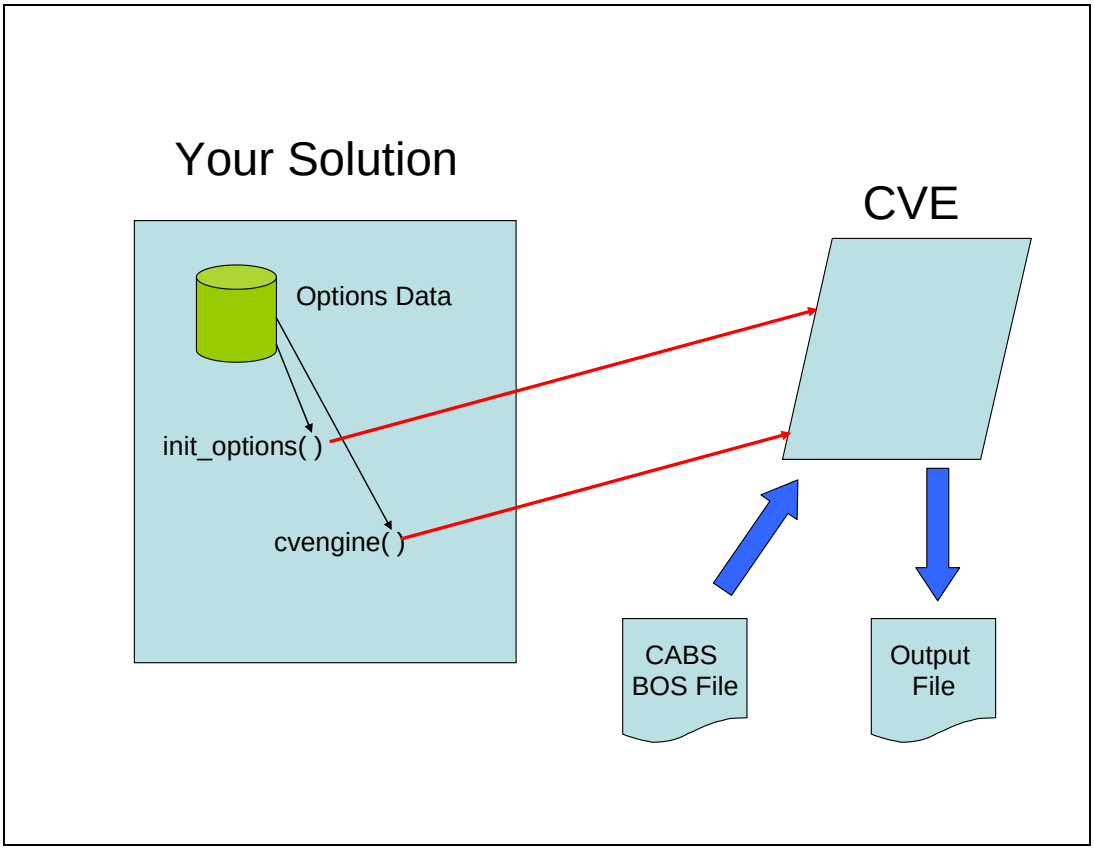

Figure 4. CABS Viewer Engine Usage

So by using only two function calls to the CABS Viewer Engine DLL or shared library, you can process all of your CABS BOS BDT data files with all the options and power of our CABS Viewer product.

CABS Viewer Engine API Details

The CABS Viewer Engine Application Programming Interface (API) is identical regardless of your development environment.

The current API consists of three library calls and a options structure, which is populated to request specific BOS file processing options.

The library calls are in ANSI C format and may be accessed with other programming languages, if correctly ported or supported in ANSI C format.

All of the API function calls prototypes and the options structure are located in either the `cvedll.h` header file or the `cve_shared.h` header file (included in the CABS Viewer Engine distribution).

The CABS Viewer Engine API Details

The API uses the following three function calls:

```
EXPORT int cvengine(char *infilename, char *outfilename,
options_t *options);

EXPORT int init_options(options_t *options);

EXPORT int dump_options(options_t *options);
```

`cvengine()` function call details

The `cvengine` function is the core of the CABS Viewer Engine API and does most of the actual work in converting the input BOS formatted file into a CABS Viewer output formatted file. The supported formats include text, csv, xml, html and db (as set in the `options` structure).

The `infilename` argument is a NULL terminated string containing a full path to the BOS input file.

The `outfilename` argument is a NULL terminated string containing a full path to the output file. This location must be writable by the process user for both file creation and file writing.

The `options` argument is a pointer to an `options` structure, which has been previously populated with the `init_options()` call. These options will be used during CABS Viewer processing.

This function returns 0 (zero) on success and the following on failure:

-1 Error encountered during BOS file processing

`init_options()` function call details

The `init_options()` function is used to populate the `options` structure with standard settings to be used by the CABS Viewer Engine.

The `options` argument is a pointer to a local `options` structure. These options should be passed to the `cvengine()` function and will be used during CABS Viewer processing.

This function returns 0 (zero) on success and the following on failure:

-1 Error encountered during structure inititalization

`dump_options()` function call details

The `dump_options()` function is used to display the contents of the current `options` structure.

The `options` argument is a pointer to an options structure, which has been previously populated with the `init_options()` call.

This function returns 0 (zero) on success and the following on failure:

-1 Error encountered during structure processing

The `options` Structure

The options structure can be used to set specific processing options, including output file format, for the `cvengine()` function call.

The options structure will be found in either the `cvedll.h` header file or the `cve_shared.h` header file.

```
typedef struct options_tag {

// output format values ================================
#define FORMAT_TEXT     0           /* -t --text */
#define FORMAT_CSV      1           /* -c --csv */
#define FORMAT_HTML     2           /* -w --html */
#define FORMAT_XML      3           /* -x --xml */
      int display_bill;         /* -1 --billrecords */
      int display_detail;       /* -3 --detailrecords */
      int display_csr;          /* -4 --csrrecords */
      int bodyonly_flag;        /* -b --bodyonly */
      int print_common;         /* -C --common */
      int debug_flag;           /* -d --debug */
      int database_flag;        /* -D --database */
      int print_recno;          /* -n --recno */
      int numeric_flag;         /* -N --number */
      int print_header;         /* -q --noheader */
      int print_raw;            /* -r --raw */
      int print_summary;        /* -s --summary */
      int print_summarydetail;  /* -S --summarydetail */
      int print_types;          /* -T --types */
      int uppercase_flag;       /* -u --uppercase */
      int print_trailer;        /* -z --notrailer */
} options_t;
```

See Appendix C for a short description of options and the CABS Viewer Users Guide for more detailed information on these options and which options are supported in each of the output formats.

Steps to CABS Viewer Engine Success

1. Install CABS Viewer Engine to your target development platform.
2. Build with the CVE test DLL or CVE test shared library with the supplied example code. Note: this does not require a license file.
3. Build with the CVE test DLL or CVE test shared library with your code. Note: this does not require a license file.
4. Request and receive a license.
5. Build with the CVE DLL or CVE shared library with the supplied example code. Note: this does require a license file to allow you to run the example code binary.
6. Build with the CVE DLL or CVE shared library with the your code. Note: this does require a license file to allow you to run the example code binary.
7. Done!

Building with CABS Viewer Engine

Instructions for building the CABS Viewer Engine (CVE) differ according to target platform and CVE form (DLL or Shared library). The following sections detail how to build using CVE, with our supplied example code and your target development environment.

Lymeware is available for CVE support or custom services to support other target platforms and development environments. Contact our sales department for more information on these opportunities.

Building CVE on Linux

Testing your GCC installation

Many of the Linux distributions (including Red Hat Enterprise Linux) have gcc and g++ preinstalled and will not need any special installation to build the included example code.

To test for this, type the following in a bash shell console:

```
gcc –v
```

If gcc is correctly installed, a message similar to the following shall be returned:

```
Reading specs from /usr/lib/gcc/i686-pc-
linux/3.4.4/specs
Configured with: /gcc/gcc-3.4.4/gcc-3.4.4-1/configure
--verbose --prefix=/usr --
exec-prefix=/usr --sysconfdir=/etc --libdir=/usr/lib -
-libexecdir=/usr/lib --man
dir=/usr/share/man --infodir=/usr/share/info --enable-
languages=c,ada,c++,d,f77,
java,objc --enable-nls --without-included-gettext --
enable-version-specific-runt
ime-libs --without-x --enable-libgcj --disable-java-
awt --with-system-zlib --ena
ble-interpreter --disable-libgcj-debug --enable-
threads=posix --enable-java-gc=b
```

```
oehm --disable-win32-registry --enable-sjlj-exceptions
--enable-hash-synchroniza
tion --enable-libstdcxx-debug : (reconfigured)
Thread model: posix
gcc version 3.4.4 (linux special) (gdc 0.12, using dmd
0.125) Page.
```

If gcc is not found using the above command, it may need to be installed. Our Linux reference platform (Red Hat Enterprise Linux 3/4/5) uses RPM (the Red Hat Package Manager) for product installation and management. Your specific Linux distribution may use other package managers. If not using Red Hat Linux and RPM then refer to your own distribution's package installation instructions.

If installing from Red Hat, perform the following tasks:

- Log in as root
- In a bash shell, run up2date (or yum if running RHEL 5 or newer)

```
up2date gcc
```

or

```
yum install gcc
```

This will install the latest version of gcc on your system.

Building with the test CVE Shared Library

To compile the test CVE shared library with our supplied example code:

```
gcc –o test_cve_dll –cve_dll.c –L. –lcve_test
```

Building with the full CVE Shared Library

To compile the full CVE shared library with our supplied example code:

```
gcc –o full_cve_dll –cve_dll.c –L. –lcvengine
```

We recommend that the CVE shared library be located with your final binary and that LD_LIBRARY_PATH be used to allow shared library discovery. For example:

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/your/bin

Building CVE on Solaris

The newest versions of Solaris have GCC preinstalled and will not need any special installation to use the included GCC scripts.

To test for this, type the following in a shell window:

```
gcc –v
```

If GCC is correctly installed, a message similar to the following shall be returned:

```
This Reading specs from /usr/lib/gcc/u6-sparc-
solaris/3.4.4/specs
Configured with: /gcc/gcc-3.4.4/gcc-3.4.4-1/configure
--verbose --prefix=/usr --
exec-prefix=/usr --sysconfdir=/etc --libdir=/usr/lib -
-libexecdir=/usr/lib --man
dir=/usr/share/man --infodir=/usr/share/info --enable-
languages=c,java --enable-nls --without-included-
gettext --enable-version-specific-runt
ime-libs --without-x --enable-libgcj --disable-java-
awt --with-system-zlib --ena
ble-interpreter --disable-libgcj-debug --enable-
threads=posix --enable-java-gc=b
oehm --enable-sjlj-exceptions --enable-hash-
synchroniza
tion --enable-libstdcxx-debug : (reconfigured)
Thread model: posix
gcc version 3.4.4 (solaris-64int) (gdc 0.12, using dmd
0.125).
```

If GCC is not found using the above command, it may need to be installed. Solaris uses the PKG (Package) utilities for product installation and management.

Lymeware uses the www.sunfreeware.org web site for Solaris pre-packaged open source products. Surf to the site, select your specific platform and Solaris version and download the binary package to your local Solaris machine.

Once downloaded, perform the following tasks:

- Log in as root
- In a ksh shell, run pkgadd with the following command:

```
pkgadd –d. <GCC_pkg_name>
```

This will install the latest version of GCC on your system.

Building with the test CVE Shared Library

To compile the test CVE shared library with our supplied example code:

```
gcc –o test_cve_dll –cve_dll.c –L. –lcve_test
```

Building with the full CVE Shared Library

To compile the full CVE shared library with our supplied example code:

```
gcc –o full_cve_dll –cve_dll.c –L. –lcvengine
```

We recommend that the CVE shared library be located in **/usr/local/lib** and that LD_LIBRARY_PATH be used to allow shared library discovery. For example:

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib

Building CVE on Windows

Lymeware recommends Microsoft Visual C++ (Reference platform was Windows XP Pro SP2 with Visual C++ 6.0 SP5) for CVE development on all Windows platforms.

Building with the CVE Dynamic Link Libraries (DLLs)

The supplied example code for the Windows target platform consists of a single Visual C++ 6.0 workspace with four separate C and C++ example projects.

Project	Description
cve_c_dll	C project that links to full CVE DLL

cve_cpp_dll	C++ project that liks to full CVE DLL
test_cve_c_dll	C project that links to test CVE DLL
test_cve_cpp_dll	C++ project that liks to test CVE DLL

For each project you will only need to clean/build/run to test each binary.
Note: the full CVE DLL binaries will require a valid license to run correctly.

We recommend that the CVE DLL should be installed in the %system-root%/system32 directory in your production application or product (e.g. C:\Windows\system32).

How to Get Help

This chapter explains how to contact Lymeware Product Support if you need assistance with your CABS Viewer product.

Scope of Support Services

Lymeware Product Support can provide assistance and information for the following:

- Installing the CABS Viewer product
- CABS Viewer product questions
- Software revisions and upgrades
- Implementing a specific feature
- How to use the CABS Viewer product
- The status of your support call
- Requests for product enhancement

Unfortunately, we cannot assist you with problems involving the following, but we may be able to suggest a next step or another vendor to call:

- Your hardware
- Your operating system or other system software
- Your application or user-written programs
- Software not developed by Lymeware Corporation
- Scripts written by Lymeware consultants, partners, or other third parties

Try this first

Before you call Lymeware Product Support, use your software manuals (including this manual) to locate the section that documents the program or feature where you are having problems. The documentation may explain the software's behaviour or give you insight to help you solve the problem.

Contact Lymeware Product Support

Two e-mail addresses are available for CABS Viewer product support or to report a potential bug in the software or documentation. Please use the following addresses:

Support@lymeware.com for all technical inquires and problem reports, including documentation issues from customers with support contracts. Customers should include relevant contact details, including company name and phone number, in initial message to speed processing. Messages that are continuations of an existing problem report should include the problem report ID in the subject line. Customers without support contracts with Lymeware Corporation should not use this address, but should contact their distributor directly.

Bugs@lymeware.com for bug reports and documentation problems.

Bug reports on software releases are always welcome. These may be sent by any means, but e-mail to the bug reporting address listed above is preferred. Please send proposed fixes and successful workarounds with the report if possible. Additional useful information would include **CABS Viewer** software version, hardware description, operating system version and patches, screen dumps, relevant sections of logs and configuration files, and failed messages files. Any reports will be acknowledged, but further action is not guaranteed. Any changes resulting from bug reports may be included in future releases.

Appendix A - CONFIGURATION WORKSHEETS AND FORMS

This appendix contains worksheets that should be used to complete specific tasks during the installation, configuration and maintenance of your CABS Viewer product. The following table describes each worksheet.

License Request Form	This form is required for issuing of an evaluation or permanent license (required for product operation)
Problem Reporting Form	This form should be used for any problems encountered which can be reported back to Lymeware

Table 1. CABS Viewer Product Worksheets

These worksheets may be copied for use in maintaining your CABS Viewer product.

License Request Form

A specific license data file will be required to run a CABS Viewer product. Lymeware or your distributor will be able to supply a valid license file if the following information is supplied:

CABS Viewer License Request Form Version 1.3
For requesting a valid commercial or evaluation CABS Viewer product license
Customer specific information
Customer (Company) Name:
Lymeware Product Name: **CABS Viewer Engine**
Lymeware Product Version:
Lymeware Product Options: **standard**
Target Machine IP Address:
Target Machine Host ID (only needed for Solaris):
Target Machine Make and Model:
Target Machine Operating System and Version:
Contact Person:
Contact Phone Number:
Contact E-mail Address:
This License Request Form may be faxed to Lymeware Corporation at (240) 218-7363 or the same information may be e-mailed to sales@lymeware.com.

The license file will be delivered to the Contact E-Mail Address. The license file must be installed in the CABS Viewer install directory as `license.dat` and must be owned by root.

CABS Viewer Problem Report Form

Version 1.2

For reporting CABS Viewer product problems
Customer specific information
Your Name:
Your Company Name:
Your Telephone Number:
Your E-mail Address:
Your CABS Viewer product and version:
Your CABS Viewer platform:
Any software add-ons to your CABS Viewer system:
A detailed description of the problem:
The sequence of steps that led to the problem:
Actions you have taken to diagnose or resolve the problem:
This Problem Report Form may be faxed to Lymeware Corporation at (240) 218-7363 or the same information may be e-mailed to sales@lymeware.com.

Appendix B – Example Code

This example code is provided in platform specific download packages.

```
// cvedll.h - CABS Viewer Engine DLL header file
//
//  Copyright (c) 2003-2007 Lymeware Corporation,  All Rights Reserved
//  THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF LYMEWARE
CORPORATION
//     The copyright notice above does not evidence any actual or
//     intended publication of such source code.
//
// $Header: /projRCS/cabsviewer/src/cvedll.h,v 1.6 2007/01/14 12:44:36 aclark Exp $
// Additional Engine DLL functions for CABS Viewer
//

#ifndef CVE_DLL_H
#define CVE_DLL_H 1

#if CPP == 1
  #define EXPORT extern "C" __declspec(dllimport)
#else
  #ifdef BUILD_DLL
    // the dll exports
    #define EXPORT __declspec(dllexport)
  #else
    // the exe imports
    #define EXPORT __declspec(dllimport)
  #endif
#endif
//#define EXPORT

// CVE types for DLL function calls ----------------------------------
//

typedef struct options_tag {
        // configuration globals ======================================
        //     externally set (config file, command line switches, etc)
        //      in switch ascii order
        int output_format;
        // output format values ======================================
#define FORMAT_TEXT         0       /* -t --text */
#define FORMAT_CSV          1       /* -c --csv */
#define FORMAT_HTML         2       /* -w --html */
#define FORMAT_XML          3       /* -x --xml */

        int display_bill;       /* -1 --billrecords */
        int display_detail;             /* -3 --detailrecords */
```

```
        int display_csr;        /* -4 --csrrecords */
        int bodyonly_flag;              /* -b --bodyonly */
                                        /* -c --csv */
        int print_common;               /* -C --common */
        int debug_flag;                 /* -d --debug */
        int database_flag;              /* -D --database */
        int print_recno;        /* -n --recno */
        int numeric_flag;               /* -N --number */
                                        /* -o FILE --output=FILE */
        int print_header;               /* -q --noheader */
        int print_raw;                  /* -r --raw */
        int print_summary;              /* -s --summary */
        int print_summarydetail; /* -S --summarydetail */
                                        /* -t --text */
        int print_types;        /* -T --types */
        int uppercase_flag;             /* -u --uppercase */
                                        /* -w --html */
                                        /* -x --xml */
        int print_trailer;      /* -z --notrailer */
} options_t;

// function to be imported/exported
EXPORT int cvengine(char *infilename, char *outfilename, options_t *options);
EXPORT int init_options(options_t *options);
EXPORT int dump_options(options_t *options);

#endif /* ifdef CVE_DLL_H */
// EOF
```

```
// cve_dll.c : Defines the entry point for the console application.

// used to test the CABS Viewer Engine DLL function

        // at this point the options can be modified directly

        // now call the main processing function with:
        //  an input file name (which must exist and be readable) and
        //  an output file name (which should not exist and will be in a
        //     writable location) and
        //  the populated options structure, which dictates the processing
        //     that should take place

        // Note: the default output format is TEXT
        cvengine("demo_data42","demo_data42.text",&opts);
        return (0);
```

```
}
// EOF
```

Appendix C – CABS Viewer Engine Options

CABS Viewer Engine supports the following Options to modify or select the exact output format and content required for your application:
The full form is:

Option Descriptions

Switch	Description
bodyonly	Display only the <BODY> section of HTML output format.
common	Print common block for each BDT record.
debug	Display debug messages to standard out (screen).
database	Display CSV records in database format.
recno	Display BDT record numbers.
number	Number the displayable field labels.
raw	Display raw BDT records.
noheader	Do not display output header.
summary	Print a BDT summary report.
summarydetail	Print a BDT summary detail report.
types	Display BDT Record type descriptors.
uppercase	Force field descriptions to UPPERCASE.
billrecords	Display only the Billing records (10-XX-XX)
detailrecords	Display only the Detail Billing records (30-XX-XX)
csrrecords	Display only the Customer Service Records (CSR) records (40-XX-XX)

Table 2. Command Line Options

CSV	Set output format to CSV (comma-delimited format).
TEXT	Set output format to ASCII text (default).
HTML	Set output format to HTML (webpage).
XML	Set output format to XML.

Table 3. Output Format Values

Appendix D – License Error Codes

The Lymeware license routine produces a purposeful ambiguous error message for all license errors. Only the license error code changes.

Error Code	Reason	Solution

Appendix E – GLOSSARY

ATIS - Alliance for Telecommunications Industry Solutions.

BDT - Billing Data Tape.

BOS - Billing Output Specifications, details the specific record and data element format of the CABS standard.

CABS – Carrier Access Billing System, a telecom carrier billing standard, supported by many of the major national and regional carriers.

CLEC – Competitive Local Exchange Carrier.

CSV – Comma Separated Values (also called command-delimited format) usually supported by spreadsheet programs and as import formats for many databases.

HTML – The Hyper Text Mark-up Language. The data format used to build and describe web pages.

ILEC – Incumbent Local Exchange Carrier.

ITU-T - International Telecommunications Union - Telecommunications standardization sector.

OSS – Operational Support System. In the Telecommunications Industry the OSS is the sum of all in-house provisioning and billing systems and databases.

RBOC – Regional Bell (system) Operating Company. The pieces of AT&T created to provide Local telephone service. Often referred to as the "Baby Bells".

URL – Universal Resource Locator, typically a web browser address or location value.

XML – A open data format, defined by WC3 to allow data interchange between differing programs and platforms.

CABS Viewer Engine User's Manual

Please send suggestions or corrections to:

support@lymeware.com
Lymeware Corporation
Box 1027
Old Lyme, CT 06371 USA
www.lymeware.com