# Implementing HTTPS in CONTENTdm 6

This is an overview for CONTENTdm server administrators who want to configure their CONTENTdm Server and Website to make use of HTTPS. While the CONTENTdm Server has supported HTTPS since version 4.x, you will need to upgrade to CONTENTdm 6.1.4 or later to use HTTPS with the CONTENTdm Website.

This information is for system administrators running CONTENTdm locally. It does not apply to users subscribed to CONTENTdm Hosting Services.

## Learn about:



## Section I Before You Begin

This document assumes experience with configuring and setting up secure web servers using HTTPS and with installing and configuring a CONTENTdm Server & Website.

You must install CONTENTdm 6.1.4 or later before you can make use of HTTPS with the CONTENTdm Website.

## Section II Overview

CONTENTdm 6 requires two distinct web site instances, the Server and the Website. Both can be configured to use HTTPS in addition to HTTP. The CONTENTdm Server is the Administration module for the CONTENTdm collection administrators, the repository for Project Client uploads, and the database & API server for the CONTENTdm Website. It is helpful to keep in mind the role the Server and Website play when configuring HTTPS:

**CONTENTdm Server**: The Server is configured to run only over HTTP by default. Since the Server is used by CONTENTdm Administrators to manage the collection data and by Project Client users to upload new records, it handles user authentication. To secure the Server logins and to support HTTPS in the CONTENTdm Website, you can add HTTPS to the bindings in IIS or add a virtual host in Apache. This HTTPS binding/virtual host is in addition to the HTTP one that is already configured. Since the Website and Server are typically installed to the same machine, the data calls the Website makes back to the Server can make use of the HTTP protocol.

**CONTENTdm Website**: The Website is the public-facing site, but also hosts the Website Configuration Tool and can be logged into directly for users to access restricted collections or items.

To use HTTPS with CONTENTdm, both the Server and the Website will be configured to run over HTTP and HTTPS. The Website will use HTTP for all anonymous traffic and will make calls back to the Server over HTTP as well. When a user logs into the Website (whether to the public-facing Login box or to the Website Configuration Tool) the site will redirect to HTTPS and all calls back to the Server at that point will be over HTTPS.

The Website makes a few cURL calls to itself but the majority of its traffic is back to the Server. For this to be handled in a secure way, the Website code needs to know the location of the Server's SSL certificate and be able to read it. This is straightforward when Server and

Website are installed to the same machine. If Server and Website have been separated, it will be necessary to export the Server's certificate and copy it to a location the Website can access.

The details of implementing HTTPS will not be described here. There are many ways to set up HTTPS and certificate creation and management will be particular to your environment. See the appendix at the end of this document for an example Apache configuration using HTTPS with the CONTENTdm Website.

## Section III Configuring CONTENTdm to Use HTTPS

Once you have set up your web server (IIS or Apache) to use the HTTPS protocol, it is necessary to configure CONTENTdm.

**CONTENTdm Server**: Configuring HTTPS for the CONTENTdm Server is straightforward and can be handled exclusively through IIS on Windows or Apache on Linux. For IIS you can add HTTPS to the bindings for the site instance and in Apache you can add an additional Virtual Host configuration to the CONTENTdm Server's *cdm.conf* file. The *cdm.conf* file should contain VirtualHost directives for HTTP and HTTPS instances.

**CONTENTdm Website**: Once you have configured IIS or Apache to use HTTPS for your CONTENTdm Website, the next step is to create the configurations pointing to your site instance's SSL certificate files. Due to the use of cURL the Website code needs to be able to locate the certificates.

Configuring the Website is handled by a command line PHP script, *manageHttps.php,* located in the Website installation's *admin* directory:

```
<CdmWebsite-dir>/admin/manageHttps.php

-h=on or off
-u=the Secure Server URL
-s=the Server SSL Certificate path
-w=the Website SSL Certificate path
```

To see help information about *manageHttps.php*, run the following from the *<CdmWebsite-dir>/admin* directory:

```
./manageHttps.php -o
```

The following series of commands could be used to enable HTTPS in a CONTENTdm Website that is installed to the same Linux machine as the CONTENTdm Server:

```
./manageHttps.php –h=on
```

The –h=on switch turns on HTTPS in the CONTENTdm code.

```
./manageHttps.php –u=your.servercertificatedomain.edu:443
```

The –u switch specifies the URL of your HTTPS CONTENTdm Server. It is not necessary to include *https://* in the URL. Be sure that the domain specified matches what was used to generate the SSL certificate. The port should always be specified, even if it is the default 443.

```
./manageHttps.php –s="/usr/local/ssl/crt/global.crt"

./manageHttps.php –w="/usr/local/ssl/crt/global.crt"
```

The –s and –w switches specified the path to the Server's and Website's SSL certificates, respectively. For an installation where Server and Website use the same domain and are distinguished only by port, the certificate can be shared. If the Server is on a different machine from the Website, the Server's certificate must be exported and made available to the Website code. In that case, the paths specified in –s and –w may not match.

 

If you are running your Website on non-standard ports (ports other than 80 for HTTP and 443 for HTTPS) you will need to add provide some additional configurations using *manageHttps.php*:

```
manageHttps.php –i=HTTP Website port

manageHttps.php –p=HTTPS Website port
```

**Windows servers:** The above examples apply to Apache on Linux. A similar set of commands would be used to configure HTTPS in IIS on Windows:

```
manageHttps.php –h=on

manageHttps.php –u=your.servercertificatedomain.edu:port

manageHttps.php –s="C:\SSL\certs\thawtePrimaryRootCA.crt"

manageHttps.php –w="C:\SSL\certs\thawtePrimaryRootCA.crt"
```

All file paths should be enclosed in quotes to account for spaces in directories or file names.

To be able to use SSL with CONTENTdm on Windows, the PHP installation needs the openSSL module to be enabled. CONTENTdm installations prior to 6.1.4 did not enable this module by default. To enable openSSL, complete the following steps:

1. Locate the php.ini file used by CONTENTdm. By default this would be located at *C:\OCLC\CONTENTdm\Content6\common\php533\php.ini*
2. Open the file in a text editor. Find the following line and remove the semicolon to uncomment the setting:

   ```
   extension=php_openssl.dll
   ```

3. Restart IIS (either manually in the IIS Manager or at the command line using iisreset).

## Section IV Testing the Setup

Once you have completed the above, verify the configuration by doing the following from a browser running on a workstation machine (not the server where CONTENTdm is installed):

1. View the Server's CONTENTdm Administration module over HTTPS:

   `https://your.servercertificatedomain.edu:port/cgi-bin/admin/start.exe`

   Log in and verify you see that Start page.

2. View the Website over HTTP:

   `http://your.website.edu:port`

   If logins are enabled, log into the Website. You should be redirected to an HTTPS login page and redirected back to an HTTPS Website if authentication is successful.

3. Log into the Website Configuration Tool:

   `https://your.website.edu/config/`

   If you try to force access to the Website Configuration Tool over HTTP, you should be redirected to the HTTPS Website instead.

 

## Appendix Example HTTPS Apache Configuration

Below is an example Apache conf file configured for HTTPS with the CONTENTdm Website. This is an internal use example where the certificate has been prepared for a fixed IP rather than a qualified domain.

```
+-----------------------------------------+
| /usr/local/Content6/Website/website.conf |
+-----------------------------------------+

# CONTENTdm Website apache configuration file
# Called from the end of httpd.conf


##############################
# The following directives are needed outside the virtual host directive
# These may be set already in http.conf so these setting may override
# the settings there.

# VIRHOST:
# Add a Listen port if you want to listen to addtional
# ports besides the one already defined in httpd.conf.
# Useful if you want virhosts on unique ports
#     THIS MUST NOT BE SET IN VIRTUAL HOST SECTION
#Listen 80

# use apache to tell php to use this ini file dir
#   THIS MUST NOT BE SET IN VIRTUAL HOST SECTION
#PHPIniDir /php

# the user and group of the web server process
# this MUST MATCH with those used in scripts/scripts.conf
#   THIS MUST NOT BE SET IN VIRTUAL HOST SECTION
User    web
#Group content   # this feature is not supported in the automated CDM installer

###############
#VIRHOST:   use as virtual host for all host names, on port80
<VirtualHost *:80>

  #VIRHOST:   uncomment and edit this to use as virtual host
  # ServerName yourserver.yourdomain.org
  # configuration for a CONTENTdm instance
  #
  # replace "/usr/local/Content6/Website/" with the full path to your installation

  # CONTENTdm needs to use these libraries
  # SetEnv LD_LIBRARY_PATH /usr/local/Content6/Website/lib:/usr/lib:/usr/local/lib

  DocumentRoot "/usr/local/Content6/Website/public_html"

  <Directory "/usr/local/Content6/Website/public_html">
    Options FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all

    SetEnv APPLICATION_ENV development

    RewriteEngine On
    #RewriteBase /
    #RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php

    RewriteCond %{REQUEST_FILENAME} -s [OR]
    RewriteCond %{REQUEST_FILENAME} -l [OR]
    RewriteCond %{REQUEST_FILENAME} -d

    # do no rewrite to anything in the oai dir
    #RewriteRule ^oai$ oai [NC,L]
```

```
    RewriteRule ^.*$ - [NC,L]
    RewriteRule ^.*$ index.php [NC,L]
  </Directory>

  DirectoryIndex index.php index.html
  AddType application/x-httpd-php .php

  <IfModule log_config_module>
        LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\""
WebsiteLogFormat
       CustomLog /usr/local/Content6/Website/../Server/logs/weblogs/WEBSITE_access.log
WebsiteLogFormat
  </IfModule>

#VIRHOST uncomment this to use as virtual host
</VirtualHost>

<IfModule mod_ssl.c>
<VirtualHost *:443>
  ServerName 132.174.11.4:443
  DocumentRoot "/usr/local/Content6/Website/public_html"

  <Directory "/usr/local/Content6/Website/public_html">
    Options FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all

    SetEnv APPLICATION_ENV development

    RewriteEngine On
    #RewriteBase /
    #RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php

    RewriteCond %{REQUEST_FILENAME} -s [OR]
    RewriteCond %{REQUEST_FILENAME} -l [OR]
    RewriteCond %{REQUEST_FILENAME} -d

    # do no rewrite to anything in the oai dir
    #RewriteRule ^oai$ oai [NC,L]

    RewriteRule ^.*$ - [NC,L]
    RewriteRule ^.*$ index.php [NC,L]
  </Directory>

  DirectoryIndex index.php index.html
  AddType application/x-httpd-php .php

  <IfModule log_config_module>
        LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\""
WebsiteLogFormat
       CustomLog /usr/local/Content6/Website/../Server/logs/weblogs/WEBSITE_access.log
WebsiteLogFormat
  </IfModule>
        #   SSL Engine Switch:
       #   Enable/Disable SSL for this virtual host.
       SSLEngine on

       #   A self-signed (snakeoil) certificate can be created by installing
       #   the ssl-cert package. See
       #   /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
       #   If both key and certificate are stored in the same file, only the
       #   SSLCertificateFile directive is needed.
       SSLCertificateFile    /etc/ssl/certs/132.174.11.4-web.crt
       SSLCertificateKeyFile /etc/ssl/private/132.174.11.4-web.key

       #   Server Certificate Chain:
       #   Point SSLCertificateChainFile at a file containing the
       #   concatenation of PEM encoded CA certificates which form the
       #   certificate chain for the server certificate. Alternatively
       #   the referenced file can be the same as SSLCertificateFile
```

 

```
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCACertificatePath /etc/ssl/certs/
#SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/apache2/ssl.crl/
#SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.  This means that
#     the standard Auth/DBMAuth methods can be used for access control.  The
#     user name is the `one line' version of the client's X.509 certificate.
#     Note that no password is obtained from the user. Every entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
```

 

```
        #   o OptRenegotiate:
        #     This enables optimized SSL connection renegotiation handling when SSL
        #     directives are used in per-directory context.
        #SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
                SSLOptions +StdEnvVars
        </Directory>

        #   SSL Protocol Adjustments:
        #   The safe and default but still SSL/TLS standard compliant shutdown
        #   approach is that mod_ssl sends the close notify alert but doesn't wait for
        #   the close notify alert from client. When you need a different shutdown
        #   approach you can use one of the following variables:
        #   o ssl-unclean-shutdown:
        #     This forces an unclean shutdown when the connection is closed, i.e. no
        #     SSL close notify alert is send or allowed to received.  This violates
        #     the SSL/TLS standard but is needed for some brain-dead browsers. Use
        #     this when you receive I/O errors because of the standard approach where
        #     mod_ssl sends the close notify alert.
        #   o ssl-accurate-shutdown:
        #     This forces an accurate shutdown when the connection is closed, i.e. a
        #     SSL close notify alert is send and mod_ssl waits for the close notify
        #     alert of the client. This is 100% SSL/TLS standard compliant, but in
        #     practice often causes hanging connections with brain-dead browsers. Use
        #     this only for browsers where you know that their SSL implementation
        #     works correctly.
        #   Notice: Most problems of broken clients are also related to the HTTP
        #   keep-alive facility, so you usually additionally want to disable
        #   keep-alive for those clients, too. Use variable "nokeepalive" for this.
        #   Similarly, one has to force some clients to use HTTP/1.0 to workaround
        #   their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
        #   "force-response-1.0" for this.
        BrowserMatch "MSIE [2-6]" \
                nokeepalive ssl-unclean-shutdown \
                downgrade-1.0 force-response-1.0
        # MSIE 7 and newer should be able to use keepalive
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
</IfModule>
```

 