## Version History

<table>
<thead>
<tr>
<th>Doc Version</th>
<th>Product</th>
<th>Date</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V6</td>
<td>OST-6.4.300</td>
<td>01/02/2015</td>
<td>English editing</td>
</tr>
</tbody>
</table>
# Table of Contents

TROUBLESHOOTING GUIDE ................................................................. III  
Table of Contents ............................................................................. v  
Table of Figures ............................................................................... vi

## CHAPTER 1: INTRODUCTION .......................................................... 1  
Abstract ............................................................................................. 1  
Scope ................................................................................................... 1

## CHAPTER 2: TECHNICAL ASPECTS .............................................. 1  
SSL Inspection overview .................................................................. 1  
  Bridge deployment ........................................................................... 1  
  UDM deployment .......................................................................... 3  
Trusted Certificates .......................................................................... 3  
  Bridge deployment ........................................................................... 3  
  UDM deployment .......................................................................... 4  
SSL Handshake Analysis (UDM mode only) ..................................... 10  
SPDY Support .................................................................................. 11  
  Prior to 6.4.300 ............................................................................ 12  
  From 6.4.300 Onwards ................................................................. 12
# Table of Figures

Figure 1: SSL Certificate Types ........................................................................................................... 2  
Figure 2: Website Security Warning Message ....................................................................................... 4  
Figure 3: CCOTTA.conf Configuration File ............................................................................................ 5  
Figure 4: Adding Certificate and Key ..................................................................................................... 6  
Figure 5: CRT File Path ............................................................................................................................ 6  
Figure 6: Internet Options ........................................................................................................................ 7  
Figure 7: Certificates Setting In IE ......................................................................................................... 7  
Figure 8: Trusted Root Certification Authorities .................................................................................... 8  
Figure 9: Selecting the Certificate from the Certificate Import Wizard ................................................ 8  
Figure 10: Adding the Certificate to the Trusted Root Certification Authorities ................................ 9  
Figure 11: Completing the Certificate Import Process ............................................................................. 9  
Figure 12: Ask User Option ..................................................................................................................... 10  
Figure 13: Invalid Certificate Message - Ask User Action .................................................................... 10  
Figure 14: Invalid Certificate Message - Blocking Action .................................................................... 10  
Figure 15: SSL Handshake High Level Overview ................................................................................ 11
Chapter 1: Introduction

Abstract
OST SSL Inspection feature, which is available in UDM deployment mode, allows for the opening up of an SSL connection (https) to analyze the content of the traffic. If this is activated, it is possible to filter HTTPS in a similar way to HTTP to enable all of the following:

- HTTPS URL analysis (not only the hostname and SNI)
- HTTPS Content analysis
- HTTPS Antivirus
- HTTPS Content analysis for Anti phishing

Scope
This document intends to give an overview about SSL Inspection technical aspects related to the solution provided by Web Safe Personal. In particular, the following aspects are described:

- Main design aspects related to OST SSL Inspection
- Considerations about OST certificate management
- Solutions for high-load environments
- Detailed description of SSL Inspection handshake

It is outside the scope of this document to discuss whether or not SSL Inspection is a suitable technology for specific architectures, and the advantages and disadvantages of SSL inspection when compared with other related techniques.

Regarding SSL handshake analysis, it is out of the scope of this section to provide full details of the messages exchanged during the process.

Chapter 2: Technical Aspects

SSL Inspection overview

Bridge deployment
WebSafe Personal can extract and analyze the certificate in order to perform actions based on the type of certificate. Possible certificates types are:

- Valid
- Invalid common name
- Invalid authority
- Self signed
- Expired
- Error
Figure 1: SSL Certificate Types

For this purpose, the SSL-connection is directly with the remote server and cannot be analyzed the content of the web site.

In this mode OST can perform the following tasks:

- **Extract and analyze the certificate** to perform actions based on the type of certificate
- **Analyze the domain name sent in the Server Name Indication (SNI)** to identify the hostname to block/bypass the site. This enables identification of different domains that share the same certificate, for example google.com and youtube.com. This is an extension of TLS protocol supported by:
  - Internet Explorer 7 or later, on Windows Vista or higher.
  - Mozilla Firefox 2.0 or later
  - Opera 8.0 (2005) or later (the TLS 1.1 protocol must be enabled)
  - Opera Mobile - at least version 10.1 beta on Android
  - Google Chrome (Vista or higher). XP on Chrome 6 or newer. OS X 10.5.7 or higher

**NOTE:** This feature does not work on Windows XP, or even Internet Explorer 8 (because the support of this feature is not browser version dependent, it depends on an SChannel system component which introduced the support of TLS SNI extension, starting from Windows Vista, not XP).
- Internet Explorer 7 or later, on Windows Vista or higher.
- Mozilla Firefox 2.0 or later
- Opera 8.0 (2005) or later (the TLS 1.1 protocol must be enabled)
- Opera Mobile - at least version 10.1 beta on Android
- Google Chrome (Vista or higher). XP on Chrome 6 or newer. OS X 10.5.7 or higher

- **Analyze the domain name of the certificate**, if no SNI is sent, to block/bypass the site. In case different domains share the same certificate, both domains will be blocked. For instance, google.com and youtube.com that share the same certificate with the domain google.com.

Note however, that working in this mode, OST cannot perform the following:

- Block by URL, only the domain name sent in the SNI. For example, the SNI, [https://www.bankex.com](https://www.bankex.com) and [https://www.bankex.com/wps/portal/int/article?WCM_GLOBAL_CONTEXT=bhint/int/helpers/contactus&proceed=1](https://www.bankex.com/wps/portal/int/article?WCM_GLOBAL_CONTEXT=bhint/int/helpers/contactus&proceed=1) are NOT the same page, but both will have the same hostname: www.bankex.com.
- Analyze the content of the site. It will be ciphered.
- Block files
- Block viruses

### UDM deployment

OST SSL Inspection in UDM mode is designed to inspect HTTPS Traffic. For SSL connections, a validated certificate is generated by the CCOTTA module. This certificate is subsequently delivered to the other end so the customer has the perception that the certificate belongs to the original server.

Based on the previous description, the SSL communication can be depicted as follows:

```
WebSafe Personal
```

This particular deployment allows CCOTTA to analyze transparently any content transmitted by either side.

In this mode OST can:
- Extract and analyze the certificate to perform actions based on the type of certificate
- Block by URL, not only by domain name (there is no sense in this mode to analyze the SNI or the domain of the certified site)
- Analyze the content of the site
- Block files
- Block viruses

### Trusted Certificates

#### Bridge deployment

OST CCOTTA module has a default list of trusted certificates in the SSL path called:
- ssl/ca_certificate.pem
- ssl/ca_certificate.crt

New trusted certificates can be added at the end of `ca_certificate.crt` file or by exporting and replacing the file with another one.

It is usually recommended to update this file only in case of a country or a company including additional trusted certificates that are not included in the standard ones.
UDM deployment

OST CCOTTA module has a default list of trusted certificates in the SSL path called:

- `ssl/ca_certificate.pem`
- `ssl/ca_certificate.crt`

New trusted certificates can be added at the end of `ca_certificate.crt` file or by exporting and replacing the file with another one.

Note however that the previous scheme does not allow third parties to provide signed certificates. Consequently, the solution is forced to generate certificates locally and this entails delivering untrusted certificates as OST is not recognized as a trusted Certificate Authority. In practice, this drawback is translated into a warning message displayed indicating a certificate exception.

![Website Security Warning Message](image)

**Figure 2: Website Security Warning Message**

The certificates sent by CCOTTA can be generated on-the-fly or imported by storing the certificate in CCOTTA server and adding to `CCOTTA.conf` the following keys:

```plaintext
https
{
  certificates
  {
    CACertificate = NEW_CERTIFICATE.crt_PATH
    CAPrivateKey = NEW_CERTIFICATE.key_PATH
  }
}
```

In the example above, the `NEW_CERTIFICATE.crt_PATH` is the path to the crt Certificate file and the `NEW_CERTIFICATE.key_PATH` is the path to the key Certificate file.

The keys should be written using the GUI following the next steps:

1. Login to the ISP GUI
2. Go to General >> Advanced Configuration and select `CCOTTA.conf` file
3. In the “Add Simple Key” text box type “https” and click on “Add Simple Key”
4. Then in “Set To List > Value” type certificates, click on add and then on set values

5. Select “certificates” from the left list of keys and again in “Set To List > Value” type:
   a. CACertificate and click on add
   b. CAPrivateKey and click on add

   If the certificates are created with password add also the following key:
   c. CAPrivateKeyPassword and click on add

6. Then click on Set Values
7. Click now on the CACertificate key in the list of keys on the left and in "Set Value" type the path of the crt file

8. The same procedure should be carried out for the CAPrivateKey (select the key from the left list) but type the path of the .key file

9. If the certificates were created with password, set the password to the key: CAPrivateKeyPassword previously added
To tackle this issue, the untrusted authority must be added to the browser CA list as a trusted authority. This allows the OST SSL Inspection solution to be easily deployed and configured at the customer side.

To add to Internet explorer the certificate, click on Tools > Internet Options

![Figure 6: Internet Options](image)

Then click on Content tab and then on “Certificates” button:

![Figure 7: Certificates Setting In IE](image)

Select “Trusted Root Certification Authorities” and click on “Import”
Click on Next and select the certificate to import used in CCOTTA:

**Figure 8: Trusted Root Certification Authorities**

Click on Next and select the option: “Place all the certificates in the following store”

**Figure 9: Selecting the Certificate from the Certificate Import Wizard**
Click on Next and then click on Finish.

In this case, the client browser will not be able to identify invalid remote server certificates so it is important to create a policy to verified invalid certificates with the action “deny” or “ask”. OST will show the client block page of invalid certificate in case a “deny” action is selected, or will ask the user to continue or not in case an “ask” action is selected.
Depending on the action selected, the browser will show a different message in case of an invalid certificate:

In case of "ask user", the following message will be displayed:

![Invalid Certificate Message - Ask User Action](image)

In case of "blocking", the following message will be displayed:

![Invalid Certificate Message - Blocking Action](image)

In case of "bypass", there will be no blocking page alerting to an untrusted certificate. Instead, the browser will load the website directly.

### SSL Handshake Analysis (UDM mode only)

This section applies only to UDM and only provides a summary of the steps that enable the SSL client and SSL server to:

- Agree on the version of the SSL protocol to use.
- Select cryptographic algorithms
- Authenticate each other by exchanging and validating digital certificates
- Use asymmetric encryption techniques to generate a shared secret key, which avoids the key distribution problem. SSL subsequently uses the shared key for the symmetric encryption of messages, which is faster than asymmetric encryption.

As mentioned in previous paragraphs, this section does not attempt to provide full details of the messages exchanged during the SSL handshake. The steps involved in the SSL handshake are as follows:

1. The SSL client sends a "client hello" message that lists cryptographic information such as the SSL version and, in the client's order of preference. The message also contains a random byte string that is used in subsequent computations. The SSL protocol allows for the "client hello" to include the data compression methods supported by the client, but current SSL implementations do not usually include this provision.

2. The SSL server responds with a "server hello" message that contains the ciphering scheme chosen by the server from the list provided by the SSL client, the session ID and another random byte string. The SSL server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).
3. The SSL client verifies the digital signature on the SSL server's digital certificate and checks that the ciphering scheme chosen by the server is acceptable.

4. The SSL client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server's public key.

5. If the SSL server sent a "client certificate request", the SSL client sends a random byte string encrypted with the client's private key, together with the client's digital certificate, or a "no digital certificate alert". This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.

6. The SSL server verifies the signature on the client certificate.

7. The SSL client sends the SSL server a "finished" message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.

8. The SSL server sends the SSL client a "finished" message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.

9. For the duration of the SSL session, the SSL server and SSL client can now exchange messages that are symmetrically encrypted with the shared secret key.

**SPDY Support**

SPDY (pronounced speedy) is an open networking protocol developed primarily at Google for transporting web content. SPDY manipulates HTTP traffic, with the particular goals of:
Reducing web page load latency: this is achieved mainly through compression (headers) and multiplexing (one connection per client).

Improving web security: SPDY requires the use of SSL/TLS (with TLS extension NPN), and does not support operation over plain HTTP.

As of July 2012, the group developing SPDY stated publicly that it is working towards standardization (available as an Internet Draft). The first draft of HTTP 2.0 uses SPDY as the working base for its specification draft and editing. HTTP 2.0 is still in draft state. In this sense the most popular internet browsers (Chromium, Mozilla Firefox, Opera and Internet Explorer) already support SPDY protocol.

The most important improvements of SPDY are:

- Multi request per connection. HTTP can only fetch one resource at a time (HTTP pipelining helps, but still enforces only a FIFO queue), a server delay of 500 ms prevents reuse of the TCP channel for additional requests. SPDY can handle multiple requests per connection.
- Not exclusively client-initiated requests. In HTTP, only the client can initiate a request. Even if the server knows the client needs a resource, SPDY can inform the client instead of waiting to receive a request for the resource from the client.
- Compressed request and response headers. Request headers today vary in size from ~200 bytes to over 2KB. As applications use more cookies and user agents expand features, typical header sizes of 700-800 bytes are common. SPDY is reducing the data in headers, thus directly improving the serialization latency to send requests.
- No redundant headers. In addition, several headers are repeatedly sent across requests on the same channel. However, headers such as the User-Agent, Host, and Accept are generally static and do not need to be resent. SPDY only sends these headers once.

Prior to 6.4.300

In versions prior to OST 6.4.300 (i.e: 6.4.200 and earlier), behavior was as follows:

- Without SSL termination activated, SPDY traffic would go through the system without being inspected, as for HTTPS.
- With SSL termination activated, SPDY would simply be disabled because we do not transmit the extensions to enable it in OST Client Hello and Server Hello. As a consequence, the browsers use normal HTTP/1.1.

From 6.4.300 Onwards

Web Safe Personal supports SPDY from software version OST 6.4.300. To detect SPDY support by the browser, OST searches in the SSL Client Hello the presence of the Next Protocol Negotiation (NPN) or Application Layer Protocol Negotiation (ALPN) extensions. If this extension is present, it is then forwarded to the server in our Client Hello:

- If the server responds with the extension activated and with SPDY protocol version 3 or 3.1, then OST announce these protocols in OST Server Hello; the client then begins the SPDY requests.
- If the server does not advertise the SPDY protocol, OST announces the client to use HTTP/1.1

Implementing SPDY layer, Web Safe Personal handles all the improvements that SPDY brings:

- Multi request per connection
- Not exclusively client-initiated requests
- Compressed request and response headers
- No redundant headers

The following chart depicts the above:

- Multi request per connection
- Not exclusively client-initiated requests
- Compressed request and response headers
- No redundant headers
- Make SSL the underlying transport protocol

CCOTTA will read the compressed header, uncompress it using zlib libraries and then send them to WebFilter.

WSP “SSL Inspection” Capability