Single Sign-On Using Enveloped Data

Document version: 4.3

May 2015

Core Platform Data Sheet

LIVEPERSON

Introduction

With LivePerson's Single Sign-On (SSO) system, visitors that have correctly logged in to your website, and then initiated a secure chat or opened the LivePerson secure mailbox, show up as being authenticated. The Agent Console then displays the correct and verified details of the authenticated visitor. SSO ensures that visitors cannot falsify their details or impersonate another visitor. Enveloped SSO is based on the PKCS #7: Cryptographic Message Syntax Standard.

Example uses of SSO include the following:

- A banking provider may want its server to automatically send the LivePerson server details about a chatting visitor (name, account number, account balance, etc.), and be certain that the visitor's details are correct.
- LivePerson customers or third-party vendors can allow visitors to simultaneously sign in to restricted areas of their website and into the LivePerson Secure Mail system.

Figure 1 below displays a high level view of the Enveloped Data model, and includes the assumption that the customer and LivePerson have already exchanged certificates.

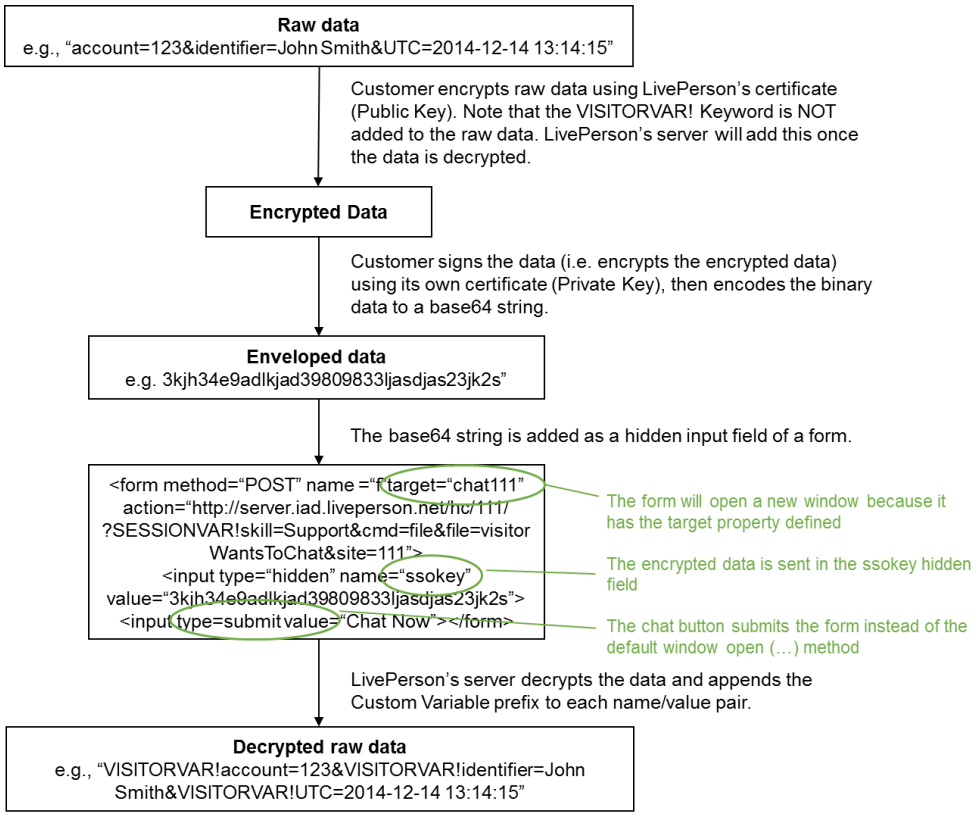

Figure 1: Enveloped Data model - High Level workflow

Prerequisites

The SSO Key preparation requires two certificates – LivePerson's public key and the customer's private key. The following steps are prerequisites for the integration:

1. Obtain LivePerson's public key certificate from the LivePerson Program Manager.
2. Send the LivePerson Program Manager the public key certificate that corresponds to the private key that will be used for signing the encrypted data. This certificate should be in Base64 encoded DER format with a `.pem` or `.cer` extension.

Single Sign-On Process

The LivePerson SSO process works as follows:

1. An authenticated visitor to the customer's website clicks the SSO-customized Chat button, or clicks the SSO-customized "Open My Secure Mailbox" button.
2. The customer's server verifies that the visitor is properly logged in to the restricted area. If the visitor is properly logged in, the customer's server collects the visitor information with the current time in UTC, formatted as yyyy-mm-dd+hh:mm:ss. This information should be set as an ampersand delimited "parameter=value" string. This string is then encrypted using the LivePerson public key, signed using the customer's private key, and converted to a Base64 string. The resulting string is commonly referred to as the ssoKey. The ssoKey can be appended to a standard LivePerson static or dynamic button, or the "Open My Secure Mailbox" button. Refer to "Code Examples" on page 5.
3. The LivePerson server processes the Chat or Mailbox request, while verifying the signature using the customer's public certificate, and decrypting the message using LivePerson's private key. Once the UTC value is validated as not older than 5 minutes, the authenticated session begins.

Notes:

I. The UTC field is mandatory.

II. A parameter called “identifier” is mandatory for chat deployments. This parameter usually holds the visitor’s full name or another unique identifier.

III. A parameter called unique_id is mandatory for mailbox deployments. This parameter is parallel to a username and usually holds an email address or other unique ID.

IV. The order of encryption and signing is not customizable.

V. By default, chats are still initiated if the SSO credential process fails, however, SSO custom variables are not shown and the authentication (Lock) icon does not appear in the Agent Console. Failed SSO chat requests can be re-routed to a different skill, an offline survey, etc., by using a rule.

VI. All data fields (key=value pairs) should be URL Encoded, including non-English Latin characters.

VII. The UTC field’s validation and permitted lifetime are configurable. Please contact LivePerson’s Technical Support to change them.

VIII. Unlike regular parameters that pass through LivePerson servers, you should not add the “VisitorVar!” variable prefix to the keys.

IX. If the "Data Encryption" option is turned on for the account, the SSO data will be encrypted at rest in LivePerson's databases. When SSO is decrypted, it will be transmitted over TLS to the agent console.

Code Examples

Chat Example – HTML

```
<html>
<body>
<form method="post"
action="https://sales.liveperson.net/hc/123456/?cmd=file&file=visitorWant
sToChat&site=123456&byhref=1" target="_blank">
<input type="hidden" name="SV!skill" value="business-ehr-english"/>
<input type="hidden" name="ssoKey"
value="MIIMkQYJKoZIhvcNAQcCoIIMgjCCDH4CAQEx… ="/>
<input type="hidden" name="SESSIONVAR!TwitterHandle" value="John123" />
<input type="hidden" name="SESSIONVAR!OriginalAgentID" value="john456" />
<input type="image"
src="https://sales.liveperson.net/hc/123456/?cmd=repstate&site=123456&ver
=1&imageUrl=https://sales.liveperson.net/hcp/Gallery/ChatButton-
Gallery/English/General/1a" alt="Click here to chat" />
</form>
</body>
</html>
```

Chat Example – JSP

```
<html>
   <body>
<input type="image"
src="https://sales.liveperson.net/hc/123456/?cmd=repstate&site=123456&ver
=1&imageUrl=https://sales.liveperson.net/hcp/Gallery/ChatButton-
Gallery/English/General/1a" alt="Click here to chat"
onclick="startSsoChat();"/>
<script type="text/javascript">
           function startSsoChat() {
               <!-- the "createSsoKey()" function should take the
parameters   you wish to send to LivePerson and encrypt & sign them to
create the SSO key dynamically -->
               var ssoKey = createSsoKey();
               if (ssoKey != null && ssoKey != '') {
                   document.getElementById("ssoKeyPostParam").value =
ssoKey;
                   document.getElementById("startSsoChatForm").submit();
               }
```

```
            }
        </script>
<form id="startSsoChatForm" name="startSsoChatForm" method="post"
action="https://sales.liveperson.net/hc/123456/?cmd=file&file=visitorWant
sToChat&site=123456&byhref=1" target="_blank">
            <input type="hidden" name="SV!skill" value="business-ehr-
english"/>
            <input type="hidden" id="ssoKeyPostParam" name="ssoKey">
        </form>
              </body>
</html>
```

Secure Mailbox Example

```
<html>
    <body>
      <form method="post" action="https://base.liveperson.net/hc/s-
      123456/web/ticket/InBoxController.jsp" target="_blank">
      <input type="hidden" name="SV!skill" value="testskill"/>
      <input type="hidden" name="ssoKey"
      value="MIIMgQYJKoZIhvcNAQcCoIIMcjCCD…."/>
      <input class=button type="submit" value="Email" name="button">
      </form>
    </body>
</html>
```

Optional SSO Implementation Methods

If the ssoKey delivery method is a GET request (appended to a URL), it should be further URLencoded. The ssoKey can be appended to a standard LivePerson static or dynamic button, or the "Open My Secure Mailbox" button.

Dynamic Buttons using mtagconfig.js

Dynamic Buttons are controlled by business rules defined in the LivePerson Admin Console, and work in conjunction with the LivePerson Monitoring Tag implemented on the page. In order to add SSO to chats initiated from these buttons, edit the mtagconfig.js file and add the SSOURL parameter to the definition of the button under the var lpMTagConfig definition, for example:

```
“dynButton” : [{“name”:”dynamic button 1”,
“afterStartPage”:true,“SSOURL”:”https://myDomain/keygen.asp”}]
```

The LivePerson server then knows to forward the button click to the keygen.asp, and to attach the visitorWantsToChat URL as a URL parameter. In keygen.asp the ssoKey should be created and the request should be forwarded to the received visitorWantsToChat URL, along with the ssoKey as a parameter.

In order to use invitations with SSO, the definition:

```
"inviteAllSSOurl": https://myDomain/keygen.asp
```

should be added to the mtagconfig.js.

Dynamic Buttons using le-mtagconfig.js

This can be configured by the account manager only.

Sending the ssoKey using a GET Request

In order to enhance its security, the ssoKey is usually transmitted using a POST request. If the ssoKey is to be delivered using a GET request by appending it to a URL, the base64 string should be further URL-encoded.

Debugging Pages

ssoKey Static Test Page

During development of the ssoKey code, it is possible to verify that a generated key can be fully decrypted by LivePerson. To check a generated ssoKey, login as administrator at:

https://<LPdomain>/hc/web/public/pub/ma/lp/login.jsp?goto=VerifyEnvelopedSSO.jsp

The LivePerson domain can be provided by your Account Manager.
In this page:

- Paste the generated ssoKey in the second part of the page.
- Choose the method by which the ssoKey will be sent by your code. If the method is GET, make sure that the ssoKey is URL-Encoded.
- Click Verify.

If the ssoKey is decrypted successfully, the original parameter string will be displayed along with the list of parameters as param=value pairs. If a test agent is online, a test chat can be initiated using the Click-to-Chat button. If errors are encountered, please refer to **Appendix A – Possible Errors** for hints on what is causing the issue.

ssoKey Error Trapping Page

Once an ssoKey is decrypted successfully in the static page, it is time to test the code that opens a chat window while sending the ssoKey to LivePerson.

Using a test agent, start a chat that utilizes your code. When clicking on the visitor that is shown in the Agent Console, there should be an icon in the SSO column. This column can be added if it is not visible. In addition to the icon, under the Info tab there are parameters displayed on the visitor, grouped under a few subjects. If the icon is visible, the SSO parameters will be listed under the "Single Sign-On" header. If the icon is not displayed, it is possible to learn what error was encountered on LivePerson's side.

Browse to:

https://<LPdomain>/hc/web/public/pub/ma/lp/login.jsp?goto=ssoErrorTracking.jsp

Log in using a user with administrator privileges. In the visitor IP field, enter the IP of the visitor that failed the SSO chat. The IP can be taken from the Info tab of that user in the Agent Console. Click "Submit" and the error that was encountered will be displayed.

Refer to Appendix A – Possible Errors for hints on what is causing the issue.

Appendix A – Possible Errors

Due to the fact that an ssoKey is a string of characters that is encrypted twice and encoded once or twice, there is a chance that, during the integration effort, the ssoKey will not be decrypted successfully by LivePerson. The decryption process is a complex mathematical formula with multiple stages. Therefore, it is usually impossible to pin-point an exact reason why the process didn't succeed, but the errors can provide a hint on possible causes.

The collection of errors and possible explanations below are an accumulation done by LivePerson as a result of issues encountered during past integrations. As noted, they should only be used as a careful pointer to a possible explanation.

Base64 Decoding Phase

In this phase, an attempt is made on the LivePerson server to decode the base64 encoded key.

- Any error that mentions DEF or DER, such as:
  - IOException converting stream to byte array: DEF length 3201 object truncated by 1
  - unknown object in factory: org.bouncycastle.asn1.DERUnknownTag
  - unknown object in factory: org.bouncycastle.asn1.DERApplicationSpecific

These errors usually mean the following:

- The base64 string is not complete – Use an online base64 decoder to check if the string can be decoded.
- The base64 string is URL encoded but the method chosen for test is POST.

Signature Verification ("Unsign") Phase

In this phase, an attempt is made on the LivePerson server, to verify the signature that was performed using the private key on the customer's server.

- Message signature was not verified Site's certificate is [0] Version: 3 SerialNumber: **12345** IssuerDN: C=US,O=ComapnyXYZ CA1 Start Date: Tue Apr 26 12:41:11 EDT 2011 Final Date: Fri Apr 26 13:11:11 EDT 2013 SubjectDN: C=US,O= ComapnyXYZ CA1,OU=NA,OU=CALC,CN=prodkey<……..>message certificate is X509CertSelector: [Serial Number: **67890** Issuer: O= ComapnyXYZ QACA1,C=US matchAllSubjectAltNames flag: true] Sign Algorithm is: SHA256

The customer's public key that is stored in LivePerson for this SiteID has serial number **12345**, while the private key that signed the ssoKey has the serial number **67890**. Either use the private key with serial number **12345** to sign the ssoKey, or send LivePerson the public key with the serial number of **67890**.

ssoKey Decryption Phase

In this phase, an attempt is made to decrypt the stream of bytes that was produced by the verification process, in order to get the original string of parameters.

- Unable to decrypt the data. LP Certificate: [[Version: V1 Subject: CN=LivePerson Inc., OU=LivePerson Inc., O=LivePerson Inc., L=New York, ST=NY, C=US <…..> Validity: [From: Wed Sep 01 12:58:26 EDT 2004, To: Tue Nov 30 11:58:26 EST 2004] Issuer: CN=LivePerson Inc., OU=LivePerson Inc., O=LivePerson Inc., L=New York, ST=NY, C=US SerialNumber:

  [**111111**]] <….> X509CertSelector: [Serial Number: **12345** Issuer: O=CompanyXYZ QACA1,C=US matchAllSubjectAltNames flag: true] X509CertSelector: [Serial Number: **222222** Issuer: CN=LivePerson Inc.,OU=LivePerson,O=LivePerson,L=New York,ST=NY,C=US matchAllSubjectAltNames flag: true]

LivePerson's public key with serial number **111111** was used unsuccessfully to decrypt the stream, instead of the public key with serial no. **222222**. Contact LivePerson Technical Support in order to correct the key used for your account.

Parameters Check Phase

In this phase, the presence of the required parameters is checked and the UTC timestamp is checked for validity.

- Missing SSO custom variable: identifier (missing FNAME from pattern {FNAME} {LNAME})

`identifier` is a mandatory parameter and it is missing. It usually contains the name of the visitor but it can be any unique string.

- Message: Unknown error, could not verify sso key

  Exception: Unparseable date: ""2011-06-11 19:53:48""

The UTC parameter value should be entered without quotes, for example: UTC=2011-06-11 19:53:48

- SSOkey UTC time is too old (2011-06-11 19:53:48)

The UTC timestamp is deviating by more than 5 minutes from the LivePerson's server clock. If this error is shown during the integration phase, it is recommended to ask LivePerson to turn off the timestamp check until the dynamic portion of the integration (using actual code to start a chat with SSO) is working.