KeyShield SSO

server API

ver. 6.1 (2. Sep. 2015)

TDP

Na Pankráci 54, Praha 4
Czech Republic, Europe

KeyShield SSO introduction

KeyShield used to be a standard IdP SSO solution. It provides 3 standard and commonly used SSO interfaces – SAML2, Radius Accounting and eDirectory Network Address Attribute management. Beside them, a proprietary but easy to implement and incredible fast REST API is available. If your system is already IdP or LDAP enabled, the KeyShield SSO integration can be completely done within a working day. The server is Java based with direct support for Linux and Windows installation, while native clients are available for all major client platforms – Windows XP and higher, Linux, MacOSX, iOS, Android, BlackBerry10.

With all advantages mentioned above, KeyShield SSO is an ideal solution when you need to make your system SSO enabled.
There are few reasons, why to use the REST API even when SAML is fully supported:

1. KeyShield SSO can support virtually any protocol used by your system – e.g. WebDAV or proprietary client communication – when you implement our REST API.
2. REST API is pretty easy to implement, you don't need to waste your valuable resources. 1 working day for the integration would fit to any schedule.
3. With KeyShield API, your solution will be much faster then with any other SSO solution. The server is able to process up to 10.000 authentication request per second. This is possible because the user is pre-authenticated via our native client. Typical user gets authenticated during workstation log-on procedure. The KeyShield SSO server is able to provide the user's identity within 100 microseconds.
4. With the user identity certificates, also multi layered systems can be supported. When the front-end layer of your system is able to communicate with the client directly, the certificate received from KeyShield SSO server can be passed to an application or database layer.
5. What is easy for you, would be easy for your customers as well. Don't worry about months or even years spent on customer's SSO projects – with KeyShield, your customers can have a productive SSO environment within a day.

KeyShield SSO – how does it work?

KeyShield SSO always uses at least one so called User Source. It can be eDirectory, Active Directory, Open LDAP, Generic LDAP or the built in ApacheDS based directory. Here you can find one of the most important advantages – KeyShield SSO can work with as many directories at the time as the customer needs. We have nice mixed environments with both eDirectory and Active Directory. If the customer has no directory or he don't want to use the existing directory for SSO purposes, it's easy to use the built-in directory – you just enable this option.

When the user source (s) is configured, users can authenticate and their identities can be provided to integrated systems.

User can authenticate manually, by entering username and password at any supported platform. On windows, NTLM and Novell client authentication is available. This is another very important advantage as the majority of users still consumes network services with windows desktops and notebooks. The functionality is described in the documentation but in short – with this feature, user needs to authenticate to network (eDirectory or Active Directory) and no further authentication to KeyShield SSO is needed.

KeyShield SSO server checks users in regular intervals, every 2 minutes by default. Thank to this, we can keep a list of the authenticated users in the memory. As mentioned above, then a response time to the SSO authentication request from an integrated system can be about 100 microseconds.

The REST API SSO request is pretty easy – an integrated application sends the user's IP address and optionally extra required attributes. The KeyShield SSO server replies with the user's identity and attributes.

In short, KeyShield SSO server tells you – I did check the identity of the user behind the IP address provided and I'm pretty sure, that before max. 2 minutes (or any time out used) he or she was here. If you use a certificate, it tells same. It's similar to common token systems – a token tells you, what was corrected at the time when it has been issued and how long you can believe to that. Our 2 minutes default time is relatively short in comparison with other systems, where 5 or even 10 minutes validity period is used.

KeyShield SSO integration security

There are many advanced settings like High Availability, Load Balancing, Include and Exclude directory parts, timeouts, etc, etc. These parameters are described within the Administrator documentation. But you don't need to care about them – they are not affecting the communication in between your system and the KeyShield SSO server.

What you need to think about are integration security related parameters. Even if you are developing the integration for a customer, who is not using any of them, it's recommended to implement them – https request to the KeyShield SSO server API interface with validation of the server's identity. This is pretty common and you will probably just let your customer's know how to get the server's certificate and how to store into the keystore on the server where your system is running.

Other option is so called API Key – if this feature is active, any integrated system has to prove itself by the API key, otherwise no information is provided. This protects valuable and sensitive information in the customer's LAN – user to IP address assignment. Because this API Key can be a part of the URL you are sending to the KeyShield SSO server when requesting the user's identity, it's very easy to implement. The only thing you need to take care about is to make this a part of the configuration and add it to the URL when active.

As we already mentioned above, the complete integration thru our REST API can be done within a working day. With such a small investment, you can offer many new SSO features to your customers.

KeyShield SSO test environment

For testing and development, the OVF installation kit is an ideal option. It's a SLES optimized instance with pre-installed KeyShield SSO server. The only thing you have to do is to provide DNS name, IP address, root password and KeyShield admin password. Of course, convenient installation is also available for both Windows and Linux servers. In comparison with the OVF installation, you have just to choose an existing or allow included installation of Java Runtime.

Once the KeyShield SSO server is running, the configuration is available thru convenient browser interface. By default, 8485 port is used for API/ management console and 9011 for User Interface. You can change these default ports during first stage of the installation mentioned above. Please, make sure that selected ports are not blocked by any firewall on the server, your workstation or the network..

What next:

1. User Source Connector

You need to create at least one user source (directory connector). The easiest way is to use some existing LDAP directory, which is a part of your lab environment. The system you are going to integrate should use same directory or same UserIDs (common UserIDs are email address, CN, sAMAccountName etc). The connector needs user account to search the directory for user objects. Due to different design of eDirectory, Active Directory and Open LDAP, automatic creation of this user is available with eDirectory only. Otherwise you have to create this user manually first.
Similar situation is with LDAP search base definition. It can be the root with

eDirectory but it must be some existing container below the root with Active Directory.

For development, so called manual authentication is very important. Keep this option enabled and set manual login attributes to accordingly to your lab LDAP directory. It can be CN with eDirectory and sAMAccountName with Active Directory. This setting doesn't affect an information provided for integrated systems. Now try "Test" button. If KeyShield can access your directory properly, then you can save and apply your connector configuration.

2. Client Interface

Each Source Connector must have at least one User Interface. If you plan to use just one user interface (enough for single directory support development), it's recommended to use 0.0.0.0 as the interface address and to specify server address for the client configuration file. Now try "Test" button. If KeyShield can bind and listen on the address/port configured, you can save and apply your user interface configuration.

3. Client Device

For any test, you need at least one user device with KeyShield SSO Client installed. This device must be able to connect to KeyShield SSO server and to your system. The client must not be behind NAT. It's recommended to run both servers and user device on same network just for easier troubleshooting. Client software is available within the Download section of your newly installed KeyShield SSO server, on www.keyshieldsso.com website or at stores (AppStore, Google Play, BlackBerry). The Client configuration can be set manually or downloaded from your KeyShield SSO server. The only important parameter within the configuration at this stage is server address and port.
Once you set correct address, the client should connect to the server immediately. This is indicated by yellow color of the client icon.

Please, keep in mind that KeyShield SSO Client indicates it's status by 3 colors:

Red – no connection to the server

Yellow – connected to server

Green – authenticated

Manual authentication mode is not default for Windows client. Please visit the Developers corner at www.keyshieldsso.com and download the KeyShield SSO Client Mode Switcher. Than switch the client on your windows workstation to the Manual mode. The client needs to be restarted.

When the client is connected to server (status yellow), you can invoke the login dialog by clicking the client icon. This is common for all platforms.
Use any existing user object for your first authentication. You just need to enter valid value for one of attributes configured for the manual authentication for the Connector (cn, sAMAccountName etc). Use just the value, no fdn specification nor special characters etc is needed.

If the username/password is correct, you should get authenticated within 1-2 seconds. The client icon should be green and you should be able to list authenticated user at the server console (use `Users → All users`). If the server IP address is 192.168.1.10 and your IP address is 192.168.1.55, then you can check your authentication by following URL with any browser:
http://192.168.1.10/api/userByIP/192.168.1.55?type=json … (=xml, = html)
This is the way your system will use when you implement the REST API SSO integration.

In case of any issues, first check the `Configuration → Summary` page at the KeyShield server console. Last error/warning is shown here within Server message field (if not visible, no error has been encountered since last restart of the server). The complete current log is available at `Logs→ View current log`. Here you can change detail level to Client or Detailed and repeat authentication attempts without restarting the server.

Please, don't hesitate to contact your local partner or us in case of any serious troubles.

KeyShield SSO UserID recommendation

Within the IdP SSO concept, UserID (a principal's unique identification) plays a crucial role. It must be unique at least company wide, but a world wide uniqueness brings important advantage when external users are supported. While KeyShield SSO is always working with at least one LDAP user source, your application doesn't need to be LDAP enabled. Of course, it's a clear advantage when you implement for example a synchronization mechanism in between your internal user database and customer's directory. But in fact, for SSO purposes, KeyShield SSO gives you the UserID and you authenticate the identified user from your database.

KeyShield SSO is always providing so called ScreenName, which should be a company unique UserID. But your system can be installed into a long time existing environment with incompatible setup of this basic attribute. It's why we strongly recommend to implement support for both alternatives, the ScreenName and the attributes:{AttrName}.

KeyShield SSO server API

KeyShield server uses HTTP to provide its API. You can find more information about SSO function above or in the KeyShield documentation.

Single Sign On (SSO)

SSO API allows querying of user id information for specified client IP address - for devices running KeyShield client.

JSON version of this API is available at:

HTTP: GET /json/userByIP/{ip}

http://172.22.1.11:8485/json/userByIP/172.22.1.203

or with optional attributes query parameter

http://172.22.1.11:8485/json/userByIP/172.22.1.203?attributes=mail,x-memberOf

You can use any LDAP attribute as long as it's allowed in KeyShield SSO Configuration. Add Optional API attributes in Connector configuration and make sure, that the corresponding KeyShield SSO manager user has read_access_rights to the requested attributes. The special x-memberOf attribute contains list of user's LDAP groups obtained by searching LDAP server for groups with member={user FDN} (which means that the member attribute contains user's FDN = user is the member of the group).

In KeyShield SSO 5.2 API authorization keys were added. When you configure API authorization key in General Configuration section, you need to include API key in API request either as key={api key} query attribute or as a HTTP header `KeyShieldSSO-APIKey`.

Example of URL with API authorization key included:

http://172.22.1.11:8485/json/userByIP/172.22.1.203?key=ZnBg4YQfQYL6c5jFUYgxF8TX4maICAID

server JSON response:

```
{
     "ipAddress" : "172.22.1.203",
     "fdn" : "cn=dummyUser,o=org",
     "screenName" : "dummyUser",
     "authType" : "L",
      "authMethod" : "L",
      "client" : null,
      "hwTokenPresent" : false,
      "authenticatedAt" : 1332851162349
      "attributes" : {
            "mail" : "admin@testdom.cz",
            "x-memberOf" : [ "cn=group1,o=org",
                                   "cn=group2,o=org" ]
      },
      "manual" : true,
}
```

`ipAddress`

The client IP address – this will be equal to 'ip' parameter of the request.

`fdn`

User source fully specified object name (e.g LDAP dn) of the authenticated user

`screenName`

UserID - ! warning ! By default, this is CN for eDirectory and sAMAccountName for Active Directory. Your customer can use virtually any attribute (see Server Console → Configuration → Authentication connectors – User ID Attribute. Thus we can't technically ensure unique values provided by this parameter. Please, discuss this within your documentation and/or use optional attributes as mentioned in the UserID recommendation section above.

`manual`

This flag is true if the KeyShield client was configured to use UID authentication mode (user enters user id manually, with no password) - it's included for backwards compatibility only, use authType instead.

`authType`

Client authentication type. Possible values:

E ... eDirectory authentication
A ... Active Directory authentication
U ... UID authentication (user id only)
L … Manual authentication (username & password)

authMethod

Client authentication method subtype. Possible values:

Active Directory

authType = A
authMethod = AD
authMethod = AD_NTLM

eDirectory
authType = E
authMethod = EDIR

Manual – (it was originally named LDAP)
authType = L
authMethod = USERNAME
authMethod = HWTOKEN

`client`
If the user was authenticated using RADIUS , the value of client field is 'RADIUS', otherwise it's null.

`hwTokenPresent`
Boolean field indicating that HW token (RFID card) of the user is currently present on client.

`authenticatedAt`
Client authentication start timestamp (in milliseconds since midnight, January 1, 1970 UTC)

`attributes`
array of LDAP attribute values requested using ?attributes={attribute list} query.

XML version of SSO API (Note: This API is deprecated, we strongly recommend to use JSON API) is available at:

HTTP: GET /xml/userByIP/{ip}

Parameter ip is the IP address of the client computer, for which you want to get user id information.
http://172.22.1.11:8485/xml/userByIP/172.22.1.203

server returns XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0_20" class="java.beans.XMLDecoder">
 <object class="cz.tdp.kshield.UserInfo">
  <void property="ipAddress">
   <string>172.22.14.2</string>
  </void>
  <void property="FDN">
   <string>cn=admin,o=org</string>
  </void>
  <void property="screenName">
   <string>admin</string>
  </void>
 </object>
  <void property="authType">
```

```
   <string>L</string>
  </void>
  <void property="authenticatedAt">
   <long>1332937007227</long>
  </void>
  <void property="attributes">
   <object class="java.util.TreeMap">
    <void method="put">
     <string>mail</string>
     <string>admin@testdom.cz</string>
    </void>
    <void method="put">
     <string>x-memberOf</string>
     <array class="java.lang.Object" length="2">
      <void index="0">
       <string>cn=group1,o=org</string>
      </void>
      <void index="1">
       <string>cn=group2,o=org</string>
      </void>
     </array>
    </void>
   </object>
  </void>
</java>
```

if there is no logged in user for the given IP address KeyShield server will return:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0_20" class="java.beans.XMLDecoder">
 <object class="cz.tdp.kshield.UserInfo"/>
</java>
```

We have provided convenient KeyShield Java library implementation - see WSTLib directory. See KSDemo directory for Spring Security integration example.

Sample perl script for getting the user ID for the given IP address:

```
use LWP::Simple;
use XML::Simple;

$url = "http://127.0.0.1:8485";
$ip_address = $ARGV[$0];

my $xs = XML::Simple->new();

# načtení XML dokumentu s UserInfo pro zadanou IP adresu
my $tree = $xs->XMLin(get ($url."/userByIP/".$ip_address));

# výpis fdn
print $tree->{ object }->{ void }[0]->{ string };
```

Authentication Certificate API

Authentication Certificate API provides a certificate, which certifies the user authentication at specified IP address. With the certificate, not only the user interface of your system but also the engine behind can validate user identity.

KeyShield SSO has own internal CA for signing these Authentication Certificates. It's why you need a keystore with valid certificate and key pair. The certificate can be self-signed or signed by the customer CA provider, it depends on the particular customer's security concept. KeyShield will work with any valid certificate but it's highly recommended to use a certificate which is intended for signing. The keystore must be a .p12 file.

The keystore can be uploaded via KeyShield server console → Config → General/Web_interface/API_keystore. The keystore password must be entered into the `API keystore password` field.

When `Optional API attributes` settings contains parameter "`mail`" and the certificate API request will ask for mail (`attributes=mail`), the issued certificate will contain the user's email address.

Certificate API is available at:

HTTP: GET /api/userByIP/{ip}

http://172.22.1.11:8485/api/userByIP/172.22.1.203

or with optional attributes query parameter

http://172.22.1.11:8485/api/userByIP/172.22.1.203?type=cer&attributes=mail

You can use special URL to request user info for IP address of current host using:

http://172.22.1.11:8485/api/userByIP/myip

When requesting userinfo for 'myip' no API key is required as the server returns user information only for current IP address.

It's possible to request JSON/XML/html (dynamic/UserInfo.ftl template) user information using /api/userByIP URI. You can specify requested response type using HTTP Accept header with following values:

Certificate:

Accept: "application/pkix-cert"

JSON:

Accept: “`application/json`”

XML:

Accept: “text/xml”

html:

Accept: “text/html”

Alternative option is to use type={cer/json/xml/html} query attribute. But it only works if requested (Accept) content type is text/html. For example when testing URL from web browser.

When user is not authenticated at specified IP address, server returns empty user info:

```
{"screenName":null,"ipAddress":"172.22.1.200","password":null,"manual":false,"au
thType":null,"authenticatedAt":0,"attributes":null,"connectorID":null,"fdn":null
}
```

In case of “application/pkix-cert”, server returns HTTP response code 204 No Content and no certificate is returned if the user is not authenticated.

User info attributes are stored as alternative subject attributes in the returned certificates. ASN.1 id's are:

- cn 2.5.4.3
- `FDN` 2.5.4.49
- `Connector ID` 2.5.4.45
- GUID `1.3.6.1.1.16.1`
- `mail as rfc822Name subjectAltName (rfc3280#section-4.2.1.7)`
- `IP as iPAddress subjectAltName`