# Table of Contents


Table of Contents 1
Single Sign On 2
- SAML Authentication 3
  - Using SAML SSO Authentication 4
  - Setting up SAML SSO Authentication 5
  - Configuring OneLogin as an Identity Provider 7
- LDAP Authentication 8
  - Using LDAP Authentication 9
  - Setting up LDAP Authentication 10
- Remote Authentication 11
  - Using Remote Authentication 12
  - Using Active Directory for Remote Authentication 14
  - Different ways for Remote Authentication 15
  - Remote Authentication scenarios 17
  - Using Auto Authentication 18
- Social Sites Authentication 19
  - Setting up Social Site Authentication 20
  - Using Social Sites Login 21

## SAML Authentication

## Using SAML SSO Authentication

Security Assertion Markup Language (SAML) is a standard for logging users into applications based on their sessions in another context. This single sign-on (SSO) login standard has significant advantages over logging in using a username and password. SAML is very powerful and flexible, but the specification can be quite a handful.

### Steps to enable SAML SSO Authentication

- See this article for details

### Tracking logins

You can see how your remote authentication works in the KBPublisher login logs, located under **Logs > Logins**.

For debugging, the most recent remote login is logged to a file called *last_remote_login.log* in the KBPublisher cache directory (*APP_CACHE_DIR* in *admin/config.inc.php*). For example: */home/username/kb_cache/last_remote_login.log*

The following steps assume that you have an account with a supported identity provider. You need to know the SAML Login URL and have the x.509 certificate supplied by your identity provider. You should also be familiar with the format of the SAML identity response from your SAML provider.

## Enabling SAML authentication

1. As a user with administrator privileges, go to **Settings → Authentication**.

2. On the **SAML** tab, check **Enable Single Sign-On**
   (make sure that $conf['auth_remote'] in the file *admin/config.inc.php* is set to 1)

3. Select an **Authentication Mode**, which defaults to *Only SAML Authentication allowed*. The available options are:
   - **Both built-in and SAML Authentication allowed** - User will be able to log in to KBPublisher using SSO and/or built-in authentication.
   - **Only SAML Authentication allowed** - User can only log in using SSO.
   - **Only SAML Authentication allowed, try auto authentication** - User can only log in using SSO and, if possible, auto-authentication on the SAML server will be applied.

4. In the **Multi-Factor Authentication** field, choose *The same as MFA Policy* setting if you want the built-in MFA to be available for SAML authentication. You can select the desired MFA policy in **Admin → Security/Privacy → Multi-Factor Authentication**.

5. Enter the following SAML Configuration settings:
   - **Name** - The name for your SSO provider, to be presented on the login page.
   - **Issuer** - Unique identifier for your Identity Provider (typically a URL). In some cases, this is called the Entity ID.
   - **Single Sign-On Service Url** - The SAML Login URL where the Controller will route Service Provider (SP)-initiated login requests. This is required.
   - **Single Logout Service Url** - The URL where the Controller will redirect users after they log out. If you do not specify a logout URL, users will get the KBPublisher login screen when they log out.
   - **Single Sign-On Service Binding** and **Single Logout Service Binding** - You can change the binding for login and logout URLs. Defaults to HTTP-Redirect. The other option is HTTP-POST.
   - **Public X.509 Certificate** - The x.509 certificate from your identity provider configuration.
   - **Authentication Contexts -** Set possible auth context values, place a value on a new line. Leave it empty for default value.
     Example:
     urn:oasis:names:tc:SAML:2.0:ac:classes:Password
     urn:oasis:names:tc:SAML:2.0:ac:classes:X509
     If empty It defaults to: *urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport*
     For Azure AD set to: *urn:oasis:names:tc:SAML:2.0:ac:classes:Password*

6. In the SAML **User Mapping Fields** settings, specify how SAML-authenticated users are identified in the KBPublisher Controller:
   - **Remote User Id** - Unique identifier for the user in the SAML response. This value is responsible to identify user in authentication requests. It defaults to the SAML NameID element.
   - **First Name** - The first name for the user corresponding to the KBPublisher First Name field. Given the sample response, this value would be User.firstName.
   - **Last name** - The last name for the user corresponding to the KBPublisher Last Name field. Given the sample response, this value would be User.lastName.
   - **Email** - The user's email address, corresponding to KBPublisher email field. The value must be unique among all SAML users. Given the sample response, this value would be User.email.
   - **Username** - This value corresponds to the KBPublisher username field. The value must be unique among all SAML users. Given the sample response below, the value for this setting would be User.email.

   SAMPLE RESPONSE

```
<saml:AttributeStatement>
  ...
  <saml:Attribute Name="User.OpenIDName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">adynamo</saml:AttributeValue>
  </saml:Attribute>
  ...
  <saml:Attribute Name="User.firstName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">John</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="User.lastName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Doo</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="User.email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
```

```
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">John.Doo@example.com</saml:AttributeValue>
  </saml:Attribute>
  ...
</saml:AttributeStatement>
```

7. To map SAML group attributes to KBPublisher privileges and roles, configure the **SAML attribute for Privileges** and **SAML attribute for Roles** settings. The settings you use depends on the structure of the SAML group attribute in the response.

   **Note:** You can skip this step. If you leave these settings empty they will never be rewritten on user login. You can assign required privileges and/or roles later in KBPublisher.

   1. Click the ellipsis button in the **SAML attribute for Privileges** field to open the Group-to-Privilege Mapping dialog box.
   2. Enter the *SAML group attribute name*, the *SAML group attribute value*, and choose a *KBPublisher privilege* to map to. In the sample response provided below, we map group User.group with value Editor.
   3. Click **Add**.
   4. When you have added all mapping rules, click **Done**.

   SAMPLE RESPONSE

```
<saml:AttributeStatement>
  ...
  <saml:Attribute Name="User.OpenIDName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">adynamo</saml:AttributeValue>
  </saml:Attribute>
  ...
  <saml:Attribute Name="User.group" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Editor</saml:AttributeValue>
  </saml:Attribute>
  ...
</saml:AttributeStatement>
```

   **Attention:** If you map SAML groups to KBPublisher privileges, all matched users will be assigned the specified privilege. If you do not have an Unlimited license and the number of allowed staff users is exceeded, the privilege will not be assigned to the user.

   Use the same steps to map **SAML attribute for Roles**.

8. Set up **Service Provider** values.
   - **Metadata** - Use this information to register the Knowledgebase with your identity provider.
   - **Public X.509 Certificate** - If your identity provider requires signing and/or encryption, copy the contents of your certificate and paste it here.
   - **Private Key** - If your identity provider requires signing and/or encryption, copy the contents of your private key and paste it here.
   - **Signing algorithm** - Select a signing method for all SAML requests.

## Additional configurations

- **Rewrite user on login** - This is the time, in seconds, to rewrite user data on login. Enter *0* to disable updates to user data on login. Enter *1* to rewrite user data on every authentication request.
- **User account info** - This field indicates whether or not the user is able to update his account info. Available options include:
  - **0** - OFF, user can't update his account info
  - **1** - ON, user can update his account info
  - **2** - AUTO, depends on other remote settings

## Configuring OneLogin as an Identity Provider

1. Log into your OneLogin account.

2. Go to the "Apps" section and then click the "Add app" button.
   In the filter box type "SAML" and select the "SAML Test Connector (IdP w/attr)" application.
   Change the display name of your app if you wish and click "Save".

3. In your KBPublisher administrator area go to Settings -> Authentication Provider -> SAML.
   Find the "Service Provider" section and open the "Metadata" window.
   On the Configuration tab of your Onelogin app, match the following settings:
   Audience: [Entity Id]
   Recipient: [Entity Id]
   ACS (Consumer) URL Validator: .*.
   ACS (Consumer) URL: [Assertion Consumer Service URL]
   Single Logout URL: [Single Logout Service URL]

   Click on the "Save" button.

4. KBPublisher settings for attributes names default to Onelogin attributes, but you might need to ensure they are equal.

5. Go to the "SSO" tab, then click on the "View Details" link at the "X.509 Certificate" field.
   Copy the certificate contents and paste it into the "Public X.509 Certificate" window in KBPublisher.
   Copy the values of "Issuer URL", "SAML 2.0 Endpoint (HTTP)" and "SLO Endpoint (HTTP)" and paste them into the corresponding KBPublisher fields as shown below.

6. If you don't already have a user on OneLogin go to the "Users" tab and add one.
   Go to the "Applications" tab on the user page and assign the application to him.

7. On your KBPublisher settings page click the "Test / Debug" button.
   If it works, you'll see a greeting message.

8. If click on "App Icon" in Onelogin (or Okta) to login to KBPublisher does not work, fill RelayState field in app settings, it should be full path to your KBPublisher installation (example: https;//domain.com/kb/)

## LDAP Authentication

## Using LDAP Authentication

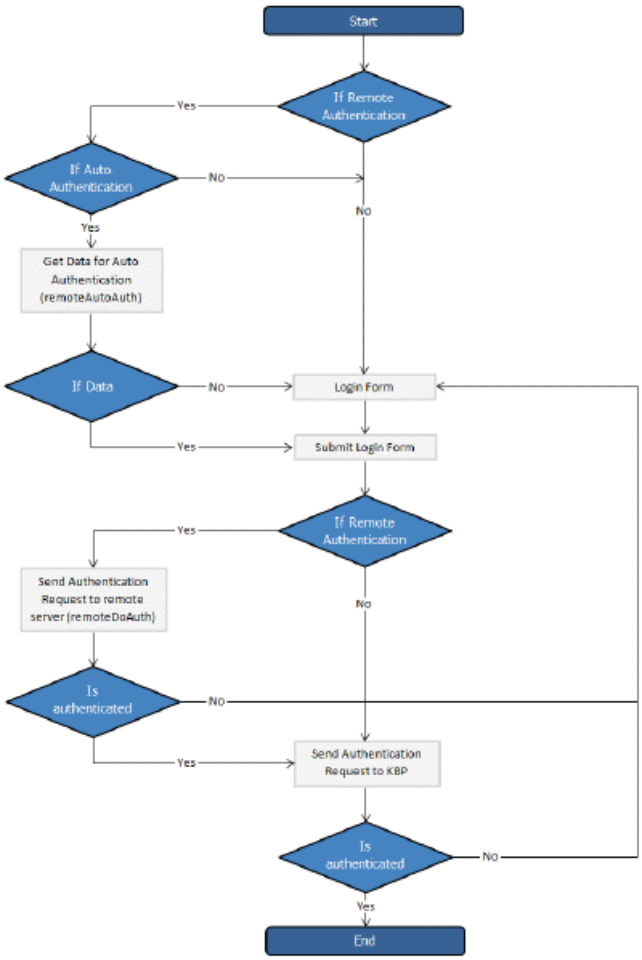
LDAP authentication allows you to integrate your organization's authentication system with KBPublisher.
Configuration requires only a few simple steps.

## Steps to enable LDAP Authentication

- See this article for details

## Quick summary of the process

- End user goes to site
- User fill the login form
- Remote Authentication checks for valid user credentials
- KBPublisher authenticates the user




## Tracking logins

You can see how your remote authentication works in logs Logs/Logins

For debugging every last login is logged to a file called *last_remote_login.log* in the KBPublisher cache directory (*APP_CACHE_DIR in admin/config.inc.php* ).
For example: */home/username/ kb_cache/last_remote_login.log*

## Setting up LDAP Authentication

LDAP Authentication is accessible by administrator in **Settings > Authentication Provider > Ldap** tab in **Admin Area**.

## Settings

- Check **Enable LDAP Authentication** checkbox.
  (make sure that $conf['auth_remote'] in the file admin/config.inc.php is set to 1)

- **LDAP Authentication**. To enable LDAP authentication check **Enable LDAP Authentication**
  Make sure that the config variable **$conf['auth_remote']** in the file admin/config.inc.php is set to 1;
  You can disable LDAP authentication by setting $conf['auth_remote'] = 0.

- **LDAP Options**. Identify host name of your LDAP server and the port for the protocols. The default port for LDAP over SSL is 636, for LDAP over TCP, UDP, or for TLS it is 389. You can specify Base DN for LDAP server and user DN and password. Leave **User DN** and **Password** fields empty for anonymous binding.

  In order to use nested groups in Active Directory, you should set the LDAP membership attribute to
  *member:1.2.840.113556.1.4.1941:*
  This is a special rule to perform a recursive group search.

  For more information, see the following:
  https://msdn.microsoft.com/en-us/library/aa746475(v=vs.85).aspx
  https://blogs.technet.microsoft.com/...explore-group-membership-with-powershell/

- **Configuration**. Set up the IP addresses for local authentication, specify whether to rewrite user data on login in "**Rewrite user on login**" field: 0 - once user created the data in KB will never been updated; 1 - on every authentication request, the user data in KB will be rewritten. You also can set up the options for updating user profiles and reseting password.

- **Group Mapping.** Here you can set up LDAP group mapping behavior. LDAP groups can be either static or dynamic. A static group entry contains a list of its users. A dynamic group is where the user entry contains a list of groups it belongs to.

- **User Mapping Fields**. Define the LDAP attributes - first name, last name, email, remote user id and attributes for privileges and roles (optional). In your expression use the round bracket to identify the specific characters to parse out the records as the field.
  For user's privilege and role mapping you can give users privileges based upon their group attributes. Click [...] button to map LDAP groups to KBPublisher Privileges and/or Roles.

  Please note, if you set mapping LDAP groups to the KBPublisher privileges, all matched users will be assigned the specified privilege. In case you do not have an Unlimited license, the number of allowed admin users could be exceeded and privilege will not be assigned to user.

  **Important!** If you leave this field blank, roles and privileges assigned for users in KBPublisher will not be updated upon login.

- **Test / Debug Options (optional)**. Set up data to test / debug authentication for actual LDAP user.

**Tip**: You can disable LDAP authentication by setting $conf['auth_remote'] = 0; in file */kbp_dir/admin/config.inc.php*

## Remote Authentication

### Using Remote Authentication

Remote authentication allows you to integrate your organization's authentication system with KBPublisher.

**Before you start:**

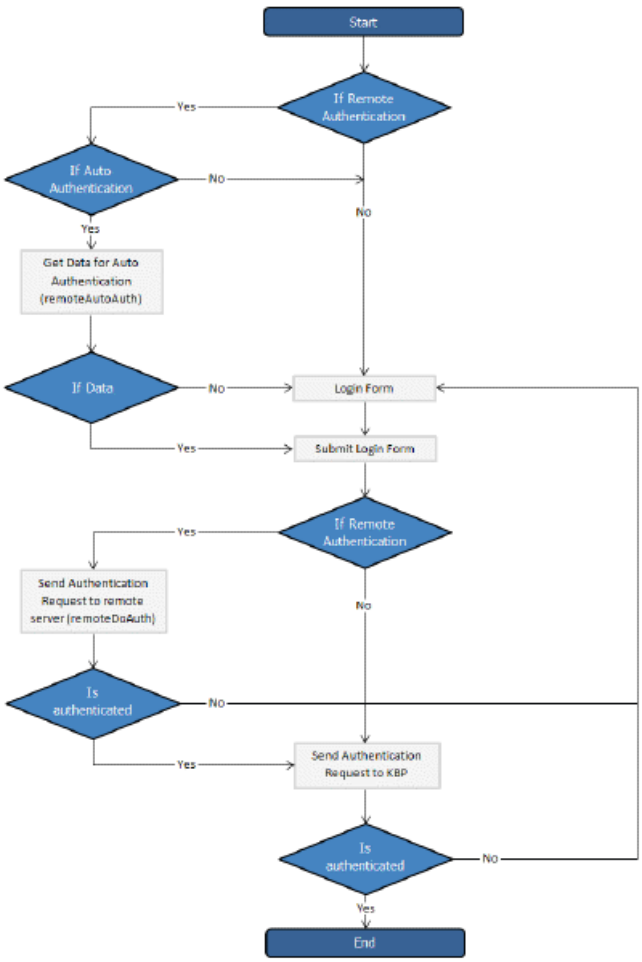
- We assume that you have some experience with PHP and with the system you are connecting to.

## Steps to enable Remote Authentication

- Click on **Settings -> Authentication Provider -> Remote**
- Check **Enable Remote Authentication** checkbox (make sure that **$conf['auth_remote']** in the file *admin/config.inc.php* is set to **1**)
- Set required values for constants in file *admin/lib/custom/remote_auth.php*
- Customize the **_remoteDoAuth** function in the file *admin/lib/custom/remote_auth.php* to authenticate the username and password passed to it against your own authentication system
- Rename the function **_remoteDoAuth** to **remoteDoAuth**

## Quick summary of the process

- End user goes to site
- Remote Authentication checks for valid user credentials
  - If auto-authentication is set, does this automatically
  - If auto-authentication is not set, user logs in first
- KBPublisher authenticates the user .




## Customizing the remoteDoAuth function

In your installation there is a folder *admin/lib/custom*. Within that folder is a file called *remote_auth.php*. This file contains the *_remoteDoAuth* function. Customize this function to do authentication against your internal system by using the username and password provided.

Here is a simple example of the function customized to authenticate against a MySQL database:

function remoteDoAuth($username, $password) {

```
    $user = false;
    $db = &DBUtil::connect($conf);

    $sql = "SELECT
        id AS 'remote_user_id',
        email, username, first_name, last_name
    FROM your_remote_users_table
    WHERE username = '%s' AND password = '%s'";
    $sql = sprintf($sql, $username, $password);
    $result = $db->Execute($sql) or die(DBUtil::error($sql, true, $db));

    // if found
    if($result->RecordCount() == 1) {
        $user = $result->FetchRow();
        $user['password'] = $password; // here you should provide not md5ing password

        // assign a priv to user (optional)
        // it is fully up to you how to determine who is authenticated and what priv to assign
        // set to off to not rewrite on login
        $user['priv_id'] = 'off';

        // assign a role to user (optional)
        // it is fully up to you how to determine who is authenticated and what role to assign
        // set to off to not rewrite on login
        $user['role_id'] = 1;
    }

    return $user;
}
```

Also see examples in attached files.

## Tracking logins

You can see how your remote authentication works in logs Logs/Logins

For debugging every last login is logged to a file called *last_remote_login.log* in the KBPublisher cache directory (*APP_CACHE_DIR in admin/config.inc.php* ).
For example: */home/username/ kb_cache/last_remote_login.log*

## Using Active Directory for Remote Authentication

It is possible to use Remote Authentication with your LDAP server.

**Before you start:**

- We assume that you have some experience with remote authentication, with PHP, and with lightweight directory access protocols (LDAP).

**Requirements**

LDAP support in PHP is not enabled by default. You will need to enable it. For more details check PHP documentation at http://php.net/ldap.

You may want to use Active Directory/PHP Helper library from http://adldap.sourceforge.net. If you do want to use it, download the library and place it into the *kb_installation_dir/admin/lib/custom* directory.

Here is an simple example of the function customized to authenticate against a LDAP server:

```
function remoteDoAuth($username, $password) {

    require_once 'custom/adLDAP.php';

    $auth = false;
    if(empty($username) || empty($password)) {
        return $auth;
    }


    //create the AD LDAP connection
    $adldap = new adLDAP();

    $user = array();
    $ldap_user = $adldap->user_info($username, array(*));

    // if found, populate $user array
    if($adldap->authenticate($username, $password)){
        $user['first_name'] = $ldap_user[0]['givenname'][0];
        ...
    }

    return $user;
}
```

You can find more examples in *kb_installation_dir/admin/lib/custom* directory.

### Different ways for Remote Authentication

There are two different types of remote authentication. It is controlled by the **KB_AUTH_TYPE** constant:

1. Adding/refreshing remote user data to KBPublisher and authenticate user.
2. Authentication by existing KBPublisher user.

## Adding/refreshing remote user data to KB and authenticate user

**KB_AUTH_TYPE = 1**

On success, the authentication function ***remoteDoAuth*** should return an associative array with the following keys:

- first_name
- last_name
- email
- username
- password-- as the user types when they login, that is, not encrypted
- remote_user_id -- a unique userID stored in your system
- role_id - (optional)
- priv_id - (optional) privilege for user. If user has a privilege, he will have access to Admin Area

## Authentication by existing KBPublisher user

**KB_AUTH_TYPE = 2**

On success, the authentication function ***remoteDoAuth*** should return.

- the user_id of the user in the KBPublisher USER table (kbp_user)
  **OR**
- Associative array with keys (user_id, username), for example: array('user_id'=>7, 'username'=>'Test').

## There are also other configuration variables

- **KB_AUTH_AREA**
  1 - Enabled for Public area only, remote authentication allowed on Public Area login screen
  2 - Enabled for Public and Admin areas

- **KB_AUTH_LOCAL**
  0 - never try to authenticate by KBPublisher built in authentication
  1 - always try to authenticate by KBPublisher built in authentication first
  2 - will try to authenticate by KBPublisher built in authentication if Remote Authentication failed

- **KB_AUTH_LOCAL_IP**
  Only users with specified IP(s) are allowed to be authenticated by KBPublisher's built in authentication
  it only matters when KB_AUTH_LOCAL = 1 or 2.
  You can set a specific IP or an IP range. Use an "-" to separate IP ranges, and a";" to separate individual IP addresses.
  For example: *127.0.0.1;210.234.12.15;192.168.1.1-192.168.255.255*

- **KB_AUTH_REFRESH_TIME**
  The time, in seconds, to rewrite user data, (3600*24*30 = 30 days), works if KB_AUTH_TYPE = 1
  0 - never. Once the user is created, data in kb table never updated by script
  1 - on every authentication request user data in the knowledgebase will be synchronized with data provided by script.

- **KB_AUTH_RESTORE_PASSWORD_LINK**
  Here you may provide a link where your remote users can restore their password.
  Set to false not to display the restore password link at all.
  KBPublisher will determine whether to set your link or the built-in one.

- **KB_AUTH_AUTO** (Using Auto Authentication)
  This variable controls whether or not the user sees a login screen and has to log in to KBPublisher, or whether they are automatically logged in.
  0 - Disabled, user gets login screen
  1 - Enabled, user doesn't see login screen
  2 - Enabled, in debug mode. User doesn't see login screen. It allows not to block "Auto Auth" if authentication failed.
  Use only for debugging and don't forget to change back to 1 or 0 when you have authentication working.

## Remote Authentication scenarios

Here are some examples/scenarios for Remote Authentication:

1. On the first authentication request the remote user is added to KBPublisher table of users.
   On the second and subsequent requests he/she is authenticated by KBPublisher's built in authentication.
   KB_AUTH_LOCAL = 1
   KB_AUTH_TYPE = 1
   KB_AUTH_REFRESH_TIME = 1.

2. Always authenticate by Remote Authentication and rewrite user data in the knowledgebase.
   KB_AUTH_LOCAL = 0
   KB_AUTH_TYPE = 1
   KB_AUTH_REFRESH_TIME = 1

3. On the first authentication request the remote user is added to KBPublisher users.
   On the second and subsequent requests the user is authenticated by remote authentication and his/her KBPublisher data is synchronized with data provided by your script, depending on the KB_AUTH_REFRESH_TIME.
   KB_AUTH_LOCAL = 0
   KB_AUTH_TYPE = 1
   KB_AUTH_REFRESH_TIME = 3600*24*30 (30 days).

4. KBPublisher tries to authenticate the user by built-in Authentication first.On failure KBPublisher tries to authenticate the user by Remote Authentication.
   KB_AUTH_LOCAL = 1
   KB_AUTH_TYPE = 2

5. If user IP matches KB_AUTH_LOCAL_IP range, then KBPublisher tries to authenticate the user by built-in Authentication first. If the IP does not match, or built-in authentication fails, KBPublisher tries to authenticate the user by Remote Authentication.
   KB_AUTH_LOCAL = 1
   KB_AUTH_LOCAL_IP = '192.168.1.1-192.168.255.255';
   KB_AUTH_TYPE = 2

Adding KB_AUTH_AUTO = 1 to any of these means the user will not be asked to type in their username and password.
System will get that information automatically. See this article how to set up Auto Authentication.

Auto authentication allows you to automatically authenticate users.

## Steps to enable Auto Authentication

- Set the constant **KB_AUTH_AUTO** in the file *admin/lib/custom/remote_auth.php* to **1**
- Customize the **_remoteAutoAuth** function in the file *admin/lib/custom/remote_auth.php* to catch current user data and return it
- Rename the function **_remoteAutoAuth** to **remoteAutoAuth**

## Customizing the remoteAutoAuth function

In your installation there is a folder *admin/lib/custom*. Within that folder is a file called *remote_auth.php*. This file contains the *_remoteAutoAuth* function. Customize this function to get the current user's credentials.

On success, the function ***remoteAutoAuth*** should return an associative array with keys (username, password) for the user. For example: array('username'=>'John', 'password'=>'Test').

Here is a simple example of the function customized using HTTP authentication:

```
function remoteAutoAuth() {

    $user = false;

    if(isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER['PHP_AUTH_PW'])) {
        $user = array();
        $user['username'] = $_SERVER['PHP_AUTH_USER'];
        $user['password'] = $_SERVER['PHP_AUTH_PW'];
    }

    return $user;
}
```

## Debugging auto-remote authentication

You can debug auto-authentication by setting **KB_AUTH_AUTO** = 2, which allows you to continually try relogging in so that you can fix any problems without having to start over every time. **Important**: Don't forget to reset this back to 1 (or 0) when you have finished debugging.

## Social Sites Authentication

## Setting up Social Site Authentication

Use Facebook, Google, Yandex, or VC authentication to log into KBPublisher, and your users will be spared from remembering another password.

The system uses the email address supplied by these services to match with an existing user in KBPublisher. If no such user is found, a new user is created.

### Enabling Social Logins

1. As a user with administrator privileges, go to **Settings > Authentication Provider**.
2. On the Social tab, check **Enable** ... checkbox for required social site.
3. Provide required ID and Secret Key for the social site.
4. Click **Save / Debug** to verify that it works.
5. Click **Save** to enable social logins for checked social sites.
6. Go to your KB login screen to test real social login.

### Where can I get social site ID and Secret Key?

See corresponding social site documentation for details.

- Google
- Facebook
- Yandex
- VK

## Using Social Sites Login

Social Sites authentication allows you to integrate social login systems with KBPublisher. Configuration requires only a few simple steps.

### Steps to enable Social Login

- See this article for details.

### Quick summary of the process

- End user goes to site.
- User clicks a button, such as "Login via [Google]".
- Popup appears with Social Site login form or, if already logged in, goes to the next step.
- Social Site Authentication checks for valid user credentials.
- KBPublisher authenticates the user.

### Tracking logins

You can see how your remote authentication works in the KBPublisher login logs, located under **Logs > Logins**.

For debugging, the most recent remote login is logged to a file called */last_remote_login.log* in the KBPublisher cache directory (*APP_CACHE_DIR* in *admin/config.inc.php*). For example: */home/username/kb_cache/last_remote_login.log*