# WAKU

Organic Data Repository documentation **0.1**

# Login

This section is about explaining all about the login procedures, that is:

- how a login should be checked
- how a user can create a login
- how an user can retrieve a password
- how the user can change his password
- how the user can change his login

## Checking the login

The first and most obvious thing to do is to validate a login, that is ask for the user to provide a login and a password, check is they are correct and give him access to various features.

For this, you need to collect from the user his login and password (most likely using a form of some sort) like this (very basic) one:

```
<form name=”LoginForm”>
   Login:<input name=”login”><br>
   Pass:<input name=”pass”><br>
   <a href=”javascript:CheckLogin()”>Log in</a>
</form>
```

Once you have collected the login and password, all you have to do is to call

```
bin/login/login.cgi?login=…&password=…&options=json
```

If the login is successfull, this should return something like

```
{"status":"ok","answer":{"valid":"...key..."}}
```

That key (ie the …key… thing) will be then sent for every subsequent calls to other functions to identify the user and his access rights.

Any other return, typically something in the form

```
{"status":"error","message":"…"}
```

implies that the user did not logged in for some reason (which is explained by the message).

All this process is likely done using such a javascript function:

```
function CheckLogin()
   {
   var Login=document.LoginForm.login.value;
   var Pass =document.LoginForm.pass.value;
   var Url  =”bin/login/login.cgi?options=json”;
   Url+=”&login=”+escape(Login);
   Url+=”&password=”+escape(Pass);
   var Answer=JSON.parse(LoadUrl(Url));
   if (Answer.status==”ok”)
      {
      alert(”Login successfull”);
      alert(”Your key is”+Answer.answer.valid);
      }
   else
      {
      alert(Answer.message);
      }
   }
```

An alternative (and somehow simpler) way to interact with that function is to call it and let it return an appropriate HTML code to display.

All you have to do is to remove the “options=json” part.

The HTML code can be configured in

```
conf/login/good.txt
```

plus a few alternative HTML page dealing with the wrong cases:

- the login/password pair doesn’t check
  `conf/login/bad.txt`
- the login is too short or wrong for some reason
  `conf/login/badlogin.txt`
- the login must be validated before being used
  `conf/login/mustvalidate.txt`
- the login has just been created (this is a special case option)
  `conf/login/user_created.txt`

## Letting users create their own login

Now let’s suppose the user doesn’t have a login yet.

How can a new login be created ?

The user will have to provide at least three basic informations:

- his intended login name
- his email
- his intended password

Note that you can reduce it to

- his email
- his password

and let the system use his email as his login: this is probably a very good solution in many cases, but be conscious that once the login is created, it cannot be changed, so if you create the login with an email address, the user won't be allowed to change his email address, he will need to create a new login with the new address.

So you likely will start with a link to "create a new login" which will go to a page containing a form like this one:

```
<form name="LoginForm">
   Login:<input name="login"><br>
   Pass:<input name="pass"><br>
   Email:<input name="email"><br>
   <a href="javascript:CreateLogin()">Create</a>
</form>
```

Then you process the form content using a function like this one:

```
function CreateLogin()
   {
   var Login=document.LoginForm.login.value;
   var Pass=document.LoginForm.pass.value;
   var email=document.LoginForm.email.value;
   var Url="bin/login/create_login.cgi?options=json";
   Url+="login="+escape(Login);
   Url+="password="+escape(Pass);
   Url+="email="+escape(Email);
   var Answer=JSON.parse(LoadUrl(Url));
   if ((Answer.status=="ok")
                    &&(Answer.answer.exist=="yes"))
      {
      alert("Login successfully created, wait mail");
      }
   else
      {
      alert(Answer.message);
      }
```

```
    }
```

It won't let you (re)create an existing login: you will recieve an error message if you try:

```
{"status":"error","message":"Login exists"}
```

Yet, it might be interesting to check if a given login does exist or not (for instance to warn the user that the login he chose is already taken). You can do so by adding "check" as an option:

```
function CheckLogin()
   {
   var Login=document.LoginForm.login.value;
   var Url="bin/login/create_login.cgi";
   Url+="?options=json+check";
   Url+="login="+escape(Login);
   var Answer=JSON.parse(LoadUrl(Url));
   if ((Answer.status=="ok")
                     &&(Answer.answer.exist=="yes"))
      {
      alert("Login exist");
      }
   else if ((Answer.status=="ok")
                     &&(Answer.answer.exist=="no"))
      {
      alert("Login is available");
      }
   else
      {
      alert(Answer.message);
      }
   }
```

Note that again, you can choose not to use json and let `create_login.cgi` return an HTML code instead (if you want to, then remove the `options=json` part of the URL).

That code can be configured in:
`conf/login/create_ok.txt`

plus the pages corresponding to the possible failures:

The login is too short or malformed
`conf/login/badlogin.txt`
The password isn't good (likely too short)
`conf/login/create_badpass.txt`
The login already exists
`conf/login/create_exist.txt`
The email address isn't provided or is malformed

```
conf/login/create_noemail.txt
```

Creating the login will create the login in an “unconfirmed” state. A large random key will be generated and the user will have to provide that key to validate the account.

The idea is that he will recieve this key by email.

## Confirming a newly created login

So, once the login is created and you recieved the answer

```
{"status":"ok","answer":{"exist"="yes"}}
```

an email will be sent to the user (to the email he just encoded). That email will ask him to follow a link to confirm his account.

The text of that email can be configured in

```
/conf/login/create_mail_text.txt
```

(the email subject is in `/conf/login/create_mail_subject.txt`)

That email should end up in calling the validation routine with the random key:

```
bin/login/confirm_login?login=…&key=…
```

You can either decide to call that function yourself (and therefore you likely will appreciate a json return) or call it straight away (and therefore you likely will appreciate some sort of html return).

In the last (and most simple) case, all you have to do is to provide the adequate HTML code in

```
conf/login/confirm_ok.txt
```

and also provide some HTML code for the various failure possibilities:

- the key is wrong
  `conf/login/confirm_fail.txt`
- the login is too short (unlikely to ever happen)
  `conf/login/confirm_logintooshort.txt`
- the login you tries to confirm doesn’t exists
  `conf/login/confirm_notexists.txt`

If you decide for the json option, then…

in your mail, you likely will have an URL pointing to your page. That URL should contain the login to validate and the random key. For instance:

```
http://myserver/mypage?key=…&login=…
```

Then you will have to extract those two data (this largely depends on your code) and call a function similar to this:

```
function ValidateLogin(Login,Key)
   {
   var Url="/bin/login/confirm_login.cgi";
   Url+="?options=json";
   Url+="login="+escape(Login);
   Url+="key="+escape(Key);
   var Answer=JSON.parse(LoadUrl(Url));
   if (Answer.status=="ok")
      {
      alert("Your login is confirmed");
      location="url of login form";
      }
   else
      {
      alert(Answer.message);
      }
   }
```

## Forgotten password

You likely will want to provide a forgot password functionality to your users.

So you need your user to enter either his login or email and he will recieve an email reminding him of his password.

So it likely all again begin with a link to "forgot password" leading to a page with a similar form:

```
<form name="LoginForm">
   Login:<input name="login"><br>
   Email:<input name="email"><br>
   <a href="javascript:Forgot()">Send my pass</a>
</form>
```

Once the user filled the form, you should call a function similar to this one:

```
function Forgot()
   {
   var Login=document.LoginForm.login.value;
   var Email=document.LoginForm.email.value;
   var Url="/bin/login/forgot_password.cgi";
   Url+="?options=json";
   Url+="login="+escape(Login);
   Url+="email="+escape(Email);
   var Answer=JSON.parse(LoadUrl(Url));
   if (Answer.status=="ok")
      {
      alert("An email has been sent");
      }
   else
      {
      alert(Answer.message);
      }
   }
```

## Loging out

Once you are logged in, you likely will want to log out sometime (note by the way that you are not forced to do so: you can just close the page and forget to log out. After a few hours your login key will not be valid anymore).

Anyhow, to log out, you can call the following code:

bin/login/logout.cgi?valid=…key…

(this is the same key that was provided by the login call).

So you likely will have a link in your user's page to the logout functionality:

```
<a href="javascript:Logout('…key…')">Log out</a>
```

that leads to such a function:

```
function Logout(Key)
   {
   var Url="bin/login/logout.cgi";
   Url+="?options=json";
   Url+="&valid="+Key;
   var Answer=JSON.parse(LoadUrl(Url));
```

```
    if (Answer.status==”ok”)
       {
       alert(”bye bye !”);
       }
    else
       {
       alert(Answer.message);
       }
    }
```

Note that a user can log in simultaneously to upto ten different sessions (for instance he can log in on his desktop computer and then on a phone).

If you use the logout this way, he will only log out from his current session.

If you want to logout from all user’s sessions you can add “all” to the options, that is:

```
function Logout(Key)
   {
   var Url=”bin/login/logout.cgi”;
   Url+=”?options=json+all”;
   Url+=”&valid=”+Key;
   var Answer=JSON.parse(LoadUrl(Url));
   if (Answer.status==”ok”)
      {
      alert(”bye bye !”);
      }
   else
      {
      alert(Answer.message);
      }
   }
```

Again, like for every function, you can skip all the JSON stuff and let the function return an appropriate HTML code configured in

```
conf/login/logout.txt
```

To do this, remove json from the options.