

Zend_Auth_Adapter_Ldap

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Auth_Adapter_Ldap Component Proposal

Proposed Component Name	Zend_Auth_Adapter_Ldap
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Auth_Adapter_Ldap
Proposers	[vincent dot dupont at ausy dot be]
Revision	1.1 - 19 March 2007: First draft. 1.2 - 03 May 2007: First revision based on prototype 1.3 - 10 May 2007: First release of code 1.4 - 24 may 2007: answers to Darby questions (wiki revision: 13)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Auth_Adapter_Ldap is a LDAP adapter for the Zend_Auth that should authenticate users from a LDAP connection.

2. References

- [PEAR::Auth](#)
- [Zend_Auth adapters poll](#)
- [PHP Native LDAP functions](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- Zend_Auth_Adapter_Ldap will **require** the LDAP extension to be loaded into PHP. This is not the case by default (see [PHP Native LDAP functions](#)). If the extension is not loaded, a Zend_Auth_Adapter_Exception will be thrown.
- Zend_Auth_Adapter_Ldap will also **require** that the LDAP directory is correctly configured and contains some data for testing. This is out of scope for the current adapter.
- Zend_Auth_Adapter_Ldap will **require** few general connection settings as the 'host'. Other settings will use default values.

- Zend_Auth_Adapter_Ldap will **require** a username and a password to be checked. If these are missing or empty, a Zend_Auth_Adapter_Exception will be thrown.
- Zend_Auth_Adapter_Ldap will only return a Zend_Auth_Result object depending on authentication success/failure (or throw Zend_Auth_Adapter_Exception if needed).
- If success, the Zend_Auth_Result Identity value will be the provided username. No complementary information about the user will be fetched from the LDAP directory (See Zend_Ldap for this)
- If failure, the Identity will be null, and the message will try to describe the error.
- Zend_Auth_Adapter_Ldap is not related in anyway to Zend_Ldap. (However, this could be a future implementation). Zend_Ldap is **not required**
- Zend_Auth_Adapter_Ldap will create a new LDAP connection on every request, and close it when finished. No connection pooling or re-use is provided.
- The process should be fast and reliable.
- Zend_Auth_Adapter_Ldap will **not save or store** the username and passwords. This means that username and password will only be used to validate the user access, and won't be recorded in any way.
- Zend_Auth_Adapter_Exception

4. Dependencies on Other Framework Components

5. Theory of Operation

The Zend_Auth_Adapter_Ldap will simply try to find the provided user login into the LDAP directory.

Using the found user information, it will try to connect and bind using the selected user information and provided password. If the bind succeeds, the authentication is ok. Otherwise, the authentication fails.

In Both cases, a Zend_Auth_Result will be returned.

If provided into the configuration settings, a bind user and password will be used for searching the user information into the LDAP directory. Otherwise, the Zend_Auth_Adapter_Ldap will try to connect anonymously.

If any technical problem occurs (LDAP not present, etc) the authentication will fail and a Zend_Auth_Adapter_Exception will be fired.

The available parameters are :

- 'host' : mandatory, the host IP or servername. Multiple hosts may be added, coma separated. This is a feature of the PHP LDAP extension : if the first host is not available, the extension will try to silently connect to the next one (PHP LDAP API feature).
- 'port' : the port of the LDAP service. Default to 389
- 'url' : alternative connection string (see ldap_connect). Will be preferred to the 'host' if provided
- 'protocol_version' : The LDAP protocol version. Default is V3
- 'bind_dn' : superuser bind username. This is used to browse the LDAP directory to get the user information
- 'bind_pw' : superuser bind password.
- 'base_dn' : base LDAP path to start searching for the user information
- 'user_dn' : user part of the LDAP path. This will be prepended to base_dn when searching for the user information (Default:"")
- 'user_oc' : the ObjectClass on which the user login will be searched for (default :posixAccount,ActiveDirectory: user)
- 'user_attr' : the attributes that correspond to the user login (default: UID, ActiveDirectory : SamAccountName)
- 'group_dn' : the DN of a group the authenticated user should be member of. If the user is authenticated against the LDAP, but is not member of the specified group, the authentication will fail (not implemented yet).

Moreover, some parameters allow special features of MS Active Directory:

- 'use_direct_bind' : use only a direct bind with the username and password (this will require a NT domain).
 - 'domain' : prepend a domain to the username
 - 'use_domain_from_email' : if the username is formatted as an email, parse the username to get the login + the domain
- The direct bind will be a faster alternative when using a Active Directory authentication.

The constructor will take 3 arguments :

- the username
 - the password provided by the enduser, and
 - the config array : holds all stable configuration information.
- This will let the developer instantiate the config array from another source like Zend_Config, and simply change the transient information, coming from the login form (username and password).

Expected Exceptions (throw a Zend_Auth_Adapter_Exception):
Constructor:

- username null or empty (message : 'Username, Password and Host must be set before calling ')
- password null or empty (message : 'Username, Password and Host must be set before calling ')
- host config null or empty (message : 'Username, Password and Host must be set before calling ')
- Ldap extension not present (message : 'Ldap extension is not properly loaded')

Authenticate():

Expected Zend_Auth_Result::FAILURE : (these are problems)

- Ldap connection failed (message : 'ldap_connect failed');FAILURE
- Ldap bind failed (anonymously or not) (message : 'ldap-bind failed') FAILURE
- Ldap search failed (search of the username into the Ldap) (Message : 'ldap_search failed') FAILURE
Expected Authentication failure (wrong user information)
- Ldap get entries failed (fetch data from Ldap)(message: 'ldap_get_entries failed') FAILURE_IDENTITY_NOT_FOUND
- Invalid username or password (message : 'Authentication failed, username or password invalid') FAILURE_CREDENTIAL_INVALID
- zero or more than 1 match for this user (message : 'ldap_get_entries return 0 or too many entries') FAILURE_IDENTITY_AMBIGUOUS

Open questions

- I figure out here that the constructor AND the authenticate() methods may throw Exceptions. If this is a problem, I can move those exceptions to the authenticate() methods. Maybe this will be more compliant?
Well I see that the Zend_Auth_Adapter_Http constructor also throw some exceptions...
- I still do not know what is the impact of authenticate() on a previously existing ldap connection. Is the existing connection reused or not?
To be tested
- Does the Adapter need to provide a storage of the authenticated Identity? I guess not but I may be missing this point in the doc.

6. Milestones / Tasks

- Milestone 1: [design notes will be published here](#)
- Milestone 2: A working prototype is currently tested. We need to add support for groups.
source at : <http://framework.zend.com/issues/browse/ZF-994>
(* Milestone 3: Working prototype checked into the incubator supporting use cases #3 and #4.)
(* Milestone 4: Unit tests exist, work, and are checked into SVN.)
(* Milestone 5: Initial documentation exists.)

If a milestone is already done, begin the description with "[DONE]", like this:

- Milestone #: [DONE] Unit tests ...

7. Class Index

- Zend_Auth_Adapter_Exception
- Zend_Auth_Adapter_Ldap

8. Use Cases

UC-01

... (see good use cases book)

9. Class Skeletons

```
class Zend_Auth_Adapter_Ldap implements Zend_Auth_Adapter_Interface {}  
  
function __construct($username, $password, $params=array())  
...
```

Here is a code snippet:

```

require_once 'Zend/Auth.php';
require_once 'Zend/Auth/Adapter/Ldap.php';

//instanciate the Zend_auth object
$auth = Zend_Auth::getInstance();

//data that should come from the login form
$username="Vincent.Dupont";
$password="xxxxxxxx";

//these are the config settings, needed to connect the the LDAP directory
$options = array(
    'host'=>'192.168.1.1', //server IP, mandatory
    'bind_dn'=>'cn=LDAPUser,OU=Users,DC=example,DC=com', //DN of the user that may
browse the ldap
    'bind_pw'=>'xxxxxxxx', //password
    'user_oc'=>'user', //ObjectClass for users
    'base_dn'=>'DC=example,DC=com', //base DN for all users
    'user_dn'=>'OU=Users', //dn for users
    'user_attr'=>'samAccountName', //the attribute that contains
the user login
    //AD options
    'use_domain_from_email'=>false, //username is an email, split
it to get the domain
    'domain'=>'', //NT domain to prepend to the
username, this is used with the direct_bind on AD
    'use_direct_bind'=>false //force the auth based only on
binding by username and password provided by user
);

try {
    // Set up the authentication adapter, this may throw an error
    $authAdapter = new Zend_Auth_Adapter_Ldap($username, $password, $options);
    // Attempt authentication, saving the result
    $result = $auth->authenticate($authAdapter);
} catch (Exception $e){
    print_r($e);
}
if (!$result->isValid()) {
    // Authentication failed; print the reasons why
    foreach ($result->getMessages() as $message) {
        echo "$message<br>";
    }
} else{
    echo 'Authentication is Ok';
    echo '<br />';
    echo 'Username is :';
    echo $auth->getIdentity();
}

```

```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>

```