

Zend_Auth_Adapter_Ldap - Michael B. Allen

<ac:macro ac:name="info"><ac:parameter ac:name="title">Zend_Auth_Adapter_Ldap Operator's Guide</ac:parameter><ac:rich-text-body>
<p>For a more practical description of how to use this adapter, please read the Zend_Auth_Adapter_Ldap Operator's Guide. This document is a more formal specification.</p></ac:rich-text-body></ac:macro>

<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	Michael B Allen Darby Felton, Zend liaison
Revision	0.1 - 5 December 2007: Created from content by Michael B Allen. 0.2 - 18 December 2007: Initial pass by Michael B Allen. 0.3 - 8 February 2008: Updated to match latest code by Michael B Allen. (wiki revision: 33)

Table of Contents

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

1. Overview

Zend_Auth_Adapter_Ldap is proposed as an authentication adapter for Zend_Auth to work with LDAP services.

2. References

- [Zend_Ldap - Michael B. Allen](#)
- The implementation on which this proposal is based is currently available here: <http://www.ioplex.com/code/>. This code implements everything described in this document.
- [The PHP LDAP API](#).
- [The now-obsolete Zend_Auth_Adapter_Ldap proposal](#).
- [RFC 4510 - Lightweight Directory Access Protocol \(LDAP\): Technical Specification Road Map](#)
- [RFC 4511 - Lightweight Directory Access Protocol \(LDAP\): The Protocol](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **must** conform to the requirements of implementations of the `Zend_Auth_Adapter` component (i.e. class `Zend_Auth_Adapter_Ldap` implements `Zend_Auth_Adapter_Interface`). Any contradiction between this proposal and implementation requirements defined by `Zend_Auth_Adapter` should be considered an error in this document.
- This component **must** use the `Zend_Ldap` component to perform all LDAP operations.
- The `Zend_Auth_Adapter::authenticate()` method **must not** return `Zend_Auth_Result::SUCCESS` unless the supplied credentials are positively validated by an LDAP server that is an authority for the account being validated.
- This component **must not** throw exceptions for conditions that may occur during normal operation with a properly configured adapter (e.g. authentication failure). All such exceptions will be caught in the adapter's `authenticate` method and translated into an appropriate `Zend_Auth_Result::FAILURE` response.
- This component **should** throw exceptions for configuration errors, environmental issues and invalid usage (e.g. required options missing, ldap extension unavailable, wrong parameter supplied to method, etc).
- This component **should** return `Zend_Auth_Result::FAILURE_IDENTITY_NOT_FOUND` if the supplied username is not found in the target LDAP servers or if the username was an empty string (AD may trigger `Zend_Auth_Result::FAILURE_CREDENTIAL_INVALID` if the supplied username is not found).
- This component **must** return `Zend_Auth_Result::FAILURE_CREDENTIAL_INVALID` if the supplied password was not correct for the supplied username or if the password was an empty string.
- This component **must** support optional SSL / TLS encrypted transport. This requirement is satisfied by `Zend_Ldap`.
- This component **must** canonicalize usernames to a form chosen by the operator (e.g. `EXAMPLE\username` or `username@example.com`). This requirement is satisfied by `Zend_Ldap`.

4. Dependencies on Other Framework Components

- `Zend_Auth_Adapter / Zend_Auth_Adapter_Interface` - This component is an implementation of the `Zend_Auth_Adapter` component and, more precisely, its `Zend_Auth_Adapter_Interface`.
- `Zend_Auth_Adapter_Exception` - The `Zend_Auth_Adapter_Exception` will be used to report configuration errors, environmental issues and invalid usage.
- `Zend_Ldap` - The bulk of the work required to authenticate credentials and canonicalize usernames is performed by the `Zend_Ldap` component.
- `Zend_Ldap_Exception` - This component catches and uses `Zend_Ldap_Exception` thrown by `Zend_Ldap`. This is a companion class to `Zend_Ldap` used to handle unexpected LDAP extension errors and LDAP specific protocol errors (e.g. failed to connect to LDAP server).

5. Theory of Operation

Because `Zend_Auth_Adapter_Interface` has one method (`authenticate()`) and because `Zend_Ldap` handles virtually all of the heavy lifting associated with validating credentials (using the `bind()` method) and canonicalizing usernames automatically during the `bind` and with the `getCanonicalAccountName()` method, the *Theory of Operation* for this class is simple.

This component iterates over an array of arrays of `Zend_Ldap` server options and attempts to bind with the supplied credentials. Note that `Zend_Ldap::bind()` will canonicalize usernames and automatically lookup the account DN if necessary (see the `Zend_Ldap` proposal for details). If the bind is successful, the canonicalized username is retrieved and `Zend_Auth_Result::SUCCESS` is returned. If the bind fails, debugging information is collected and the next set of server options is tried. If no server successfully validate the credentials, one of `Zend_Auth_Result::FAILURE_IDENTITY_NOT_FOUND`, `Zend_Auth_Result::FAILURE_CREDENTIAL_INVALID` or `Zend_Auth_Result::FAILURE` is returned with any error messages from the last iteration of the loop.

With multiple sets of server options, the adapter can authenticate users in multiple domains and provide failover so that if one server is not available, the next one will be queried.

General `Zend_Auth_Adapter` usage is covered elsewhere but its use with `Zend_Auth_Adapter_Ldap` is best summarized with the following minimalistic example:

```

$options = array(
    'server1' => array(
        'host' => 'dcl.w.net',
        'useSsl' => true,
        'accountDomainName' => 'w.net',
        'accountDomainNameShort' => 'W',
        'accountCanonicalForm' => 3,
        'baseDn' => 'CN=Users,DC=w,DC=net',
    ),
    'server2' => array(
        'host' => 's0.foo.net',
        ...
    ),
);

$username = $this->_request->getParam('username');
$password = $this->_request->getParam('password');

$auth = Zend_Auth::getInstance();

require_once 'Zend/Auth/Adapter/Ldap.php';
$adapter = new Zend_Auth_Adapter_Ldap($options, $username, $password);

$result = $auth->authenticate($adapter);

```

The names of servers (e.g. 'server1' and 'server2' shown above) are largely arbitrary.

The specific options permitted are described in the `Zend_Ldap` proposal.

The Username and Password Parameters

The second and third constructor parameters are the username and password being authenticated (e.g. the credentials supplied by the user through an HTML login form). These parameters are optional but if they are not specified they must be set using the `setUsername()` and `setPassword()` methods. A `Zend_Auth_Adapter_Exception` will be thrown if the adapter is invoked without a username and password.

The Authenticate Method

The work of a `Zend_Auth_Adapter` implementation is performed almost entirely in its `authenticate()` method.

To validate credentials with a specific server the adapter performs an LDAP "bind" operation. Normally a bind would only be performed by LDAP clients that wish to perform directory operations. However, when using LDAP as an authentication service, the act of binding itself is used to validate a username and password. Note that a password must be supplied because some LDAP servers will accept an empty password as an "anonymous bind". As per the Requirements section, this adapter will return `Zend_Auth_Result::FAILURE_CREDENTIAL_INVALID` if an empty password is supplied.

The `Zend_Ldap::bind()` method will automatically resolve the account DN and canonicalize usernames if necessary. This leaves little for the `authenticate()` method to do other than catch expected exceptions (e.g. `Zend_Ldap_Exception::LDAP_INVALID_CREDENTIALS`), record informative messages and build an appropriate result object. See the [Zend_Ldap](#) documentation for details.

The messages returned by the `Zend_Auth_Result::getMessages()` method on the object returned by the `authenticate()` method are described as follows:

Messages Array Index	String Description
Index 0	An overall message that is suitable for display to users (e.g. "Invalid credentials" or "Password required"). If the authentication was successful, array index 0 will contain an empty string.
Index 1	A detailed error message that is not suitable for display to users but is ideal for informing server operators as to the cause of the authentication failure. If the authentication was successful, array index 1 will contain an empty string.

Indexes 2 and higher	Additional messages collected in chronological order during the authentication process. The final message will contain the string at index 1 if present.
----------------------	--

All passwords will be replaced with "*****" in these messages.

6. Milestones / Tasks

- Milestone 1: [DONE] Create initial prototype.
- Milestone 2: [DONE] Create documentation necessary to use and test prototype.
- Milestone 3: Working prototype checked into the incubator.
- Milestone 4: Create unit tests

7. Class Index

- `Zend_Auth_Adapter_Ldap`

8. Use Cases

See examples herein and the [Zend_Font - Karol Babioch#5. Theory of Operation](#) section of this proposal and the [Operator's Guide](#).

9. Class Skeletons

```
class Zend_Auth_Adapter_Ldap implements Zend_Auth_Adapter_Interface
{
    public function __construct($options, $username = null, $password = null);
    public function setUsername($username);
    public function setPassword($password);
    public function getLdap();
    public function authenticate();
}
```

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