

Documentation for
mod_auth(nz)_ibmdb2
db2-auth-udfs
scripts

Helmut K. C. Tessarek

January 13, 2008

Abstract

mod_auth(nz)_ibmdb2 is an Apache authentication module using IBM[®] DB2[®] as the backend database for storing user and group information. The module supports several encryption methods.

<http://mod-auth-ibmdb2.sourceforge.net>

RCSfile: documentation.tex,v Date: 2008/01/13 16:37:44 Revision: 1.9

Contents

1	mod_auth(nz)_ibmdb2	1
1.1	Building mod_auth(nz)_ibmdb2	1
1.2	Details on building mod_auth(nz)_ibmdb2	1
1.3	Building a static module (Apache 1.3.x)	3
1.4	Installing the Manpages	3
1.5	Description of the module	4
1.6	Examples	7
2	db2-auth-udfs	9
2.1	Building the library and registering the UDFs	9
2.2	Description of the UDFs	9
3	scripts	10
3.1	Description of the scripts	10
3.2	Examples	10
4	CVS access	12
5	FAQ	13
6	Links	14
6.1	Official mod_auth(nz)_ibmdb2 website	14
6.2	Support Requests	14
6.3	Feature Requests	14
6.4	Bugs	14
6.5	Release / Update Announcement Mailing List	14
6.6	developerWorks article	14
A	directives and default values	15
B	UDF reference	16
B.1	md5	16
B.2	apr_md5	17
B.3	apr_crypt	18
B.4	apr_sha1	19
C	Stored Procedure Support	20
C.1	user authentication	21
C.2	group authentication	22

1 mod_auth(nz)_ibmdb2

1.1 Building mod_auth(nz)_ibmdb2

Log in as root user.

Change the DB2PATH and the APXS variables in the makemod script according to your environment:

```
# Path settings
```

```
DB2PATH=/home/db2inst1/sqllib  
APXS=/usr/local/apache/bin/apxs
```

Set DB2PATH to the directory where DB2 is accessed. This is usually the instance home directory.

Set APXS to the path that points to your apxs binary. The apxs binary is usually installed in the /<your apache home>/bin directory.

After changing the above settings, run the script ./makemod

Last but not least the DB2 environment has to be set in the Apache startscript. This is done by sourcing the db2profile script, which is located in DB2PATH.

1.2 Details on building mod_auth(nz)_ibmdb2

To build the module the Apache utility apxs is used. The EXTRA_LFLAG needs to be specified so that the module will find the db2 library during runtime.

```
DB2PATH=/home/db2inst1/sqllib
```

```
EXTRA_LFLAG="-Wl,-G,-blibpath:$DB2PATH/lib"      (for AIX)  
EXTRA_LFLAG="-Wl,-rpath,$DB2PATH/lib"          (for Linux)
```

```
mod_authnz_ibmdb2:
```

```
apxs -c -ldb2 $EXTRA_LFLAG mod_authnz_ibmdb2.c
```

```
mod_auth_ibmdb2:
```

```
apxs -c -ldb2 $EXTRA_LFLAG mod_auth_ibmdb2.c
```

mod_auth_ibmdb2 (Apache 1.x):

```
apxs -c -ldb2 -lcrypt -lgdbm $EXTRA_LFLAG mod_auth_ibmdb2.c
```

If the `sqlcli1.h` header file cannot be found, add the `-I` option to specify the directory where `sqlcli1.h` can be found. If the db2 library cannot be found, add the `-L` option to specify the directory where `libdb2.so` can be found.

For example:

```
apxs -c -L/home/db2inst1/sqllib/lib          \  
      -I/home/db2inst1/sqllib/include -ldb2  \  
      $EXTRA_LFLAG mod_authnz_ibmdb2.c
```

After creating the module, it has to be moved to the Apache module directory. This is also done with the `apxs` utility:

mod_authnz_ibmdb2:

```
apxs -i mod_authnz_ibmdb2.la
```

mod_auth_ibmdb2:

```
apxs -i mod_auth_ibmdb2.la
```

mod_auth_ibmdb2 (Apache 1.x):

```
apxs -i mod_auth_ibmdb2.so
```

As the next step the DB2 environment has to be set in the Apache startscript. This is done by sourcing the `db2profile` script, which is located in `DB2PATH`.

Finally, add the following directive to your `httpd.conf` and restart Apache:

mod_authnz_ibmdb2:

```
LoadModule authnz_ibmdb2_module modules/mod_authnz_ibmdb2.so
```

mod_auth_ibmdb2:

```
LoadModule ibmdb2_auth_module  modules/mod_auth_ibmdb2.so
```

mod_auth_ibmdb2 (Apache 1.x):

```
LoadModule ibmdb2_auth_module  modules/mod_auth_ibmdb2.so  
AddModule  mod_auth_ibmdb2.c
```

1.3 Building a static module (Apache 1.3.x)

Extract the Apache source code to a directory (e.g. /ext). Change into the `modules` directory of the Apache source tree:

```
cd /ext/apache_X.Y.ZZ/src/modules
```

Extract `mod_auth_ibmdb2-x.y.z.tar.gz` into the `modules` directory. Change into the `mod_auth_ibmdb2` directory and copy the two files from the `static` directory into the current one:

```
cd mod_auth_ibmdb2
cp static/.indent.pro .
cp static/Makefile.tmpl .
```

Change into the `src` directory:

```
cd ../..
```

Add the following line to the `Configuration` file and to the `Configuration.tmpl` file:

```
AddModule modules/mod_auth_ibmdb2/mod_auth_ibmdb2.o
```

Source the `db2profile`:

```
./home/db2inst1/sqllib/db2profile
```

Make the server by running `make` in the `src` directory. If you run `httpd -l` after the executable has been built, you should see `mod_auth_ibmdb2.c` in the list of compiled in modules.

1.4 Installing the Manpages

There is a `man` directory in the path, where you have extracted the `mod_auth_ibmdb2` package.

Change into the `man` directory and run the script `./maninstall`

1.5 Description of the module

mod_auth_ibmdb2 is an Apache authentication module using IBM DB2 as the backend database for storing user and group information. The module is designed for Apache 2.0.x and 1.x and supports several encryption methods. *mod_authnz_ibmdb2* is designed for Apache 2.2.x and is based on the new authentication/authorization framework.

Here is a list of the new directives¹ that come with the module:

<code>AuthIBMDB2User</code>	user for connecting to the DB2 database (no default)
<code>AuthIBMDB2Password</code>	password for connecting to the DB2 database (no default)
<code>AuthIBMDB2Database</code>	database name (no default)
<code>AuthIBMDB2UserTable</code>	name of the user table (no default)
<code>AuthIBMDB2GroupTable</code>	name of the group table (no default)
<code>AuthIBMDB2NameField</code>	name of the user field within the table (defaults to <code>username</code>)
<code>AuthIBMDB2GroupField</code>	name of the group field within the table (defaults to <code>groupname</code>)
<code>AuthIBMDB2PasswordField</code>	name of the password field within the table (defaults to <code>password</code>)
<code>AuthIBMDB2CryptedPasswords</code>	passwords are stored encrypted (defaults to <code>yes</code>)
<code>AuthIBMDB2KeepAlive</code>	connection kept open across requests (defaults to <code>yes</code>)
<code>AuthIBMDB2Authoritative</code>	lookup is authoritative (defaults to <code>yes</code>)
<code>AuthIBMDB2NoPasswd</code>	just check, if user is in usertable (defaults to <code>no</code>)
<code>AuthIBMDB2UserCondition</code>	restrict result set (no default)
<code>AuthIBMDB2GroupCondition</code>	restrict result set (no default)
<code>AuthIBMDB2UserProc</code>	stored procedure ² for user authentication (no default)
<code>AuthIBMDB2GroupProc</code>	stored procedure ² for group authentication (no default)
<code>AuthIBMDB2Caching</code>	user credentials are cached (defaults to <code>off</code>)
<code>AuthIBMDB2GroupCaching</code>	group information is cached (defaults to <code>off</code>)
<code>AuthIBMDB2CacheFile</code>	path to cache file (defaults to <code>/tmp/auth_cred_cache</code>)
<code>AuthIBMDB2CacheLifetime</code>	cache lifetime in seconds (defaults to 300)

¹see Appendix A

²SP support is not available in mod_auth_ibmdb2 for Apache 1.x, see Appendix C

If `AuthIBMDB2Authoritative` is `Off`, then iff the user is not found in the database, let other authentication modules try to find the user. Default is `On`.

If `AuthIBMDB2KeepAlive` is `On`, then the server instance will keep the IBM DB2 server connection open. In this case, the first time the connection is made, it will use the current set of `Host`, `User`, and `Password` settings. Subsequent changes to these will not affect this server, so they should all be the same in every `htaccess` file. If you need to access multiple IBM DB2 servers for this authorization scheme from the same web server, then keep this setting `Off` – this will open a new connection to the server every time it needs one. The values of the database and various tables and fields are always used from the current `.htaccess` file settings.

If `AuthIBMDB2NoPasswd` is `On`, then any password the user enters will be accepted as long as the user exists in the database.

Setting this also overrides the setting for `AuthIBMDB2PasswordField` to be the same as `AuthIBMDB2NameField` (so that the SQL statements still work when there is no password at all in the database, and to remain backward-compatible with the default values for these fields.)

For groups, we use the same `AuthIBMDB2NameField` as above for the user ID, and `AuthIBMDB2GroupField` to specify the group name.

`AuthIBMDB2GroupTable` specifies the table to use to get the group info. It defaults to the value of `AuthIBMDB2UserTable`. If you are not using groups, you do not need a `groupname` field in your database, obviously.

The optional directives `AuthIBMDB2UserCondition` and `AuthIBMDB2GroupCondition` can be used to restrict queries made against the `User` and `Group` tables. The value for each of these should be a string that you want added to the end of the where-clause when querying each table. For example, if your user table has an `active` integer field and you only want users to be able to login, if that field is 1, you could use a directive like this:

```
AuthIBMDB2UserCondition active=1
```

If `AuthIBMDB2UserProc` is set, the named stored procedure³ is responsible for returning the password of the user in question to the module. It must return exact one value - the password. If set, `AuthIBMDB2UserTable`, `AuthIBMDB2NameField`, `AuthIBMDB2PasswordField`, `AuthIBMDB2UserCondition` are ignored. If `AuthIBMDB2NoPasswd` is `On`, then the username has to be returned instead of the password. The stored procedure must have the following parameter format:

```
CREATE PROCEDURE user_procedure_name ( IN VARCHAR, OUT VARCHAR )
```

If `AuthIBMDB2GroupProc` is set, the named stored procedure⁴ is responsible for returning the groups the user in question belongs to. It must return an open cursor to the resultset. If set, `AuthIBMDB2GroupTable`, `AuthIBMDB2NameField`, `AuthIBMDB2GroupField`, `AuthIBMDB2GroupCondition` are ignored. The stored procedure must have the following parameter format:

```
CREATE PROCEDURE group_procedure_name ( IN VARCHAR )
```

If `AuthIBMDB2Caching` is set to `On`, the user credentials are cached in a file defined in `AuthIBMDB2CacheFile` and expires after `AuthIBMDB2CacheLifetime` seconds.

If `AuthIBMDB2GroupCaching` is set to `On`, the group information is cached in a cache file that is named like the file specified in `AuthIBMDB2CacheFile` but with the extension `.grp`. The cache expires after `AuthIBMDB2CacheLifetime` seconds.

³see Appendix C.1

⁴see Appendix C.2

1.6 Examples

First create the two tables within DB2:

```
CREATE TABLE WEB.USERS (
    USERNAME VARCHAR(40) NOT NULL,
    PASSWORD VARCHAR(40) );
```

```
ALTER TABLE WEB.USERS
    ADD PRIMARY KEY (USERNAME);
```

```
CREATE TABLE WEB.GROUPS (
    USERNAME VARCHAR(40) NOT NULL,
    GROUPNAME VARCHAR(40) NOT NULL );
```

```
ALTER TABLE WEB.GROUPS
    ADD PRIMARY KEY (USERNAME, GROUPNAME);
```

Then you will have to insert records into the two tables:

```
INSERT INTO WEB.USERS (username, password)
    VALUES ('test', apr_md5('testpwd'));
INSERT INTO WEB.GROUPS (username, groupname)
    VALUES ('test', 'admin');
```

Then add the following lines to your httpd.conf:

```
<Directory "/var/www/my_test_dir">
    AuthName                "DB2 Authentication"
    AuthType                 Basic
    AuthBasicProvider        ibmdb2    # Apache 2.2.x only

    AuthIBMDB2User          db2inst1
    AuthIBMDB2Password       ibmdb2
    AuthIBMDB2Database       auth
    AuthIBMDB2UserTable      web.users
    AuthIBMDB2NameField      username
    AuthIBMDB2PasswordField  passwd

    AuthIBMDB2CryptedPasswords On
    AuthIBMDB2KeepAlive      On
    AuthIBMDB2Authoritative  On
```

`apr_md5` is a User Defined Function that is explained in the `db2-auth-udfs` part of this documentation.

```
AuthIBMDB2NoPasswd      Off

AuthIBMDB2GroupTable    web.groups
AuthIBMDB2GroupField    groupname

require                  group admin
AllowOverride            None
</Directory>
```

If you want to use stored procedures and caching, the directives would look like this:

```
<Directory "/var/www/my_test_dir">
  AuthName                "DB2 Authentication"
  AuthType                Basic
  AuthBasicProvider        ibmdb2      # Apache 2.2.x only

  AuthIBMDB2User          db2inst1
  AuthIBMDB2Password      ibmdb2
  AuthIBMDB2Database      auth
  AuthIBMDB2UserProc      user_sp
  AuthIBMDB2GroupProc     group_sp

  AuthIBMDB2Caching       On
  AuthIBMDB2GroupCaching  On

  require                  group admin
  AllowOverride            None
</Directory>
```

2 db2-auth-udfs

2.1 Building the library and registering the UDFs

Login as the instance user. Change the `DB2PATH` variable in the `makeudf` script for your environment.

```
DB2PATH=/home/db2inst1/sqllib
```

Set `DB2PATH` to the directory where DB2 is accessed. This is usually the instance home directory.

After changing the above setting, start the script

```
Linux and AIX    ./makeudf
Win32           makeudf.bat
```

The UDFs are written in ANSI C and should compile on all platforms. You can use the `bldrtn` script in your `sqllib/samples/c` directory as a good start. The only thing that you have to do is to install APR and APR-util. You can get APR and APR-util at <http://apr.apache.org/>. Furthermore you need to add the compiler and linker flags for APR (see `makeudf`).

To register the UDFs, connect to your database and run the script:

```
db2 -tvf reg_udfs.ddl
```

2.2 Description of the UDFs

This UDF library delivers four functions⁵:

```
md5
apr_md5
apr_crypt
apr_sha1
```

The `md5` function is compatible to the PHP `md5` function.

The `apr_md5`, `apr_crypt` and `apr_sha1` functions are compatible to the Apache functions that are used in the `htpasswd` utility.

Note: In win32 environments `apr_crypt` returns the output of `apr_md5`.

⁵see Appendix B for a reference of the UDFs

3 scripts

3.1 Description of the scripts

There are four scripts to import the users and groups from already existing user and/or group files into DB2. They are written in php, so you should have the php cli binary in your `/usr/local/bin` directory.

The script `sync_pwds` is for syncing the system users with a table within your DB2 database.

You have to change the settings in the `config.php` file for your environment.

Here is a table of the relation between the directives for the `mod_auth(nz)_ibmdb2` module and the settings in the `config.php` file:

<code>config.php</code>		module directive
<code>\$dbname</code>	<code>= "auth";</code>	<code>AuthIBMDB2Database</code>
<code>\$dbuser</code>	<code>= "db2inst1";</code>	<code>AuthIBMDB2User</code>
<code>\$dbpwd</code>	<code>= "db2inst1";</code>	<code>AuthIBMDB2Password</code>
<code>\$usertable</code>	<code>= "users";</code>	<code>AuthIBMDB2UserTable</code>
<code>\$grouptable</code>	<code>= "groups";</code>	<code>AuthIBMDB2GroupTable</code>
<code>\$namefield</code>	<code>= "username";</code>	<code>AuthIBMDB2NameField</code>
<code>\$passwordfield</code>	<code>= "password";</code>	<code>AuthIBMDB2PasswordField</code>
<code>\$groupfield</code>	<code>= "groupname";</code>	<code>AuthIBMDB2GroupField</code>

Attention: The scripts were developed on Linux, therefore they will only work on systems where the `/etc/passwd`, the `/etc/shadow`, the `/etc/group` and the `/etc/gshadow` are in the same form as on Linux systems.

Note: `user_imp` and `group_imp` will work on all systems, because these scripts don't rely on above mentioned files.

3.2 Examples

If the settings in the `config.php` are as above and you execute the `./user_etc_imp` script following happens:

All users (except system users like root or mail) are imported from the linux box into the table `users` in the database `auth`. The table `users` has `username` as the columnname for the users and `password` as the columnname for the passwords.

To import the users from an existing htpasswd users file, just run the script

```
./user_imp <path-to-userfile>
```

To import the group information from an existing Apache group file, run the script

```
./group_imp <path-to-groupfile>
```

4 CVS access

The mod_auth(nz)_ibmdb2 CVS repository can be checked out through anonymous (pserver) CVS with the following instruction set. When prompted for a password for anonymous, simply press the Enter key.

```
cvsv -d:pserver:anonymous@evermeet.cx:/cvs login
cvsv -z3 -d:pserver:anonymous@evermeet.cx:/cvs co mod_auth_ibmdb2/module
```

where module is either mod_authnz_ibmdb2 or mod_auth_ibmdb2.

You can browse the CVS repository via the web at
http://www.evermeet.cx/cvs/mod_auth_ibmdb2

5 FAQ

Q: IBM's Websphere plugin and mod_auth(nz)_ibmdb2 seem to break each other. What can I do?

A: mod_auth(nz)_ibmdb2 has to be loaded after the Websphere plugin.

Q: Which versions of DB2 are supported?

A: All DB2 Universal Database versions currently supported by IBM. I've tested the module with DB2 UDB v7.x, DB2 UDB v8.x and DB2 9 (9.1.x and 9.5.x). Older versions should work as well.

Q: What is the difference between mod_auth_ibmdb2 and mod_authnz_ibmdb2?

A: mod_authnz_ibmdb2 is based on the new authentication backend provider scheme of Apache 2.2. This module will only work for Apache 2.2.x. mod_auth_ibmdb2 works for Apache 2.0.x and 1.x.

Q: What platforms are supported?

A: All POSIX platforms. I've compiled and tested the module on Linux and IBM AIX. Since the modules are using the APR libraries now, they can be compiled on Windows as well.

Q: Why isn't there a binary release for?

A: I don't have a development environment for every operating system. Furthermore I don't think that binary releases make sense for Unix style operating systems.

Q: What is the package db2-auth-udfs for?

A: This package contains User Defined Functions to generate passwords within DB2.

Q: How do I get support?

A: Please submit support requests at the Support Request Tracker (hosted by sourceforge).

Q: How do I submit bugs?

A: Please submit bugs at the Bug Tracker (hosted by sourceforge).

Q: How do I submit a feature request?

A: Please submit feature requests at the Feature Request Tracker (hosted by sourceforge).

6 Links

6.1 Official mod_auth(nz)_ibmdb2 website

<http://mod-auth-ibmdb2.sourceforge.net>

6.2 Support Requests

http://sourceforge.net/tracker/?atid=633718&group_id=103064&func=browse

6.3 Feature Requests

http://sourceforge.net/tracker/?atid=633720&group_id=103064&func=browse

6.4 Bugs

http://sourceforge.net/tracker/?atid=633717&group_id=103064&func=browse

6.5 Release / Update Announcement Mailing List

<http://lists.sourceforge.net/lists/subscribe/mod-auth-ibmdb2-announce>

6.6 developerWorks article

mod_auth_ibmdb2: A novel authentication method for Apache

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0407tessarek/>

A directives and default values

directive	default value
AuthIBMDB2User	–
AuthIBMDB2Password	–
AuthIBMDB2Database	–
AuthIBMDB2UserTable	–
AuthIBMDB2GroupTable	–
AuthIBMDB2NameField	username
AuthIBMDB2GroupField	groupname
AuthIBMDB2PasswordField	password
AuthIBMDB2CryptedPasswords	yes
AuthIBMDB2KeepAlive	yes
AuthIBMDB2Authoritative	yes
AuthIBMDB2NoPasswd	no
AuthIBMDB2UserCondition	–
AuthIBMDB2GroupCondition	–
AuthIBMDB2UserProc	–
AuthIBMDB2GroupProc	–
AuthIBMDB2Caching	off
AuthIBMDB2GroupCaching	off
AuthIBMDB2CacheFile	/tmp/auth_cred_cache
AuthIBMDB2CacheLifetime	300

B UDF reference

B.1 md5

```
>>-MD5--(--expression--)------><
```

MD5 hash. The `md5` function is compatible to the PHP `md5` function.

The argument can be a character string that is either a `CHAR` or `VARCHAR` not exceeding 120 bytes.

The result of the function is `VARCHAR(32)`. The result can be null; if the argument is null, the result is the null value.

Examples:

1)

```
INSERT INTO USERS (username, password)
VALUES ('test', md5('testpwd'))
```

2)

```
SELECT md5( 'testpwd' ) FROM SYSIBM.SYSDUMMY1
```

```
1
-----
342df5b036b2f28184536820af6d1caf
```

```
1 record(s) selected.
```

B.2 apr_md5

>>-APR_MD5--(--expression--)-><

Seeded MD5 hash. The `apr_md5` function is compatible to the Apache function that is used in the `htpasswd` utility.

The argument can be a character string that is either a `CHAR` or `VARCHAR` not exceeding 120 bytes.

The result of the function is `VARCHAR(37)`. The result can be null; if the argument is null, the result is the null value.

Examples:

1)

```
INSERT INTO USERS (username, password)
VALUES ('test', apr_md5('testpwd'))
```

2)

```
SELECT apr_md5( 'testpwd' ) FROM SYSIBM.SYSDUMMY1
```

```
1
```

```
-----
$apr1$HsTNH...$bmlPUSoPOF/Qhzn1.sAq6/
```

```
1 record(s) selected.
```

B.3 apr_crypt

```
>>-APR_CRYPT--(--expression--)-><
```

Unix crypt. The `apr_crypt` is compatible to the Apache function that is used in the `htpasswd` utility.

The argument can be a character string that is either a `CHAR` or `VARCHAR` not exceeding 120 bytes.

The result of the function is `VARCHAR(13)`. The result can be null; if the argument is null, the result is the null value.

Examples:

```
1)
   INSERT INTO USERS (username, password)
     VALUES ('test', apr_crypt('testpwd'))

2)
   SELECT apr_crypt( 'testpwd' ) FROM SYSIBM.SYSDUMMY1

   1
   -----
   cqs7u0vz8KB1k

   1 record(s) selected.
```

B.4 apr_sha1

>>-APR_SHA1--(--expression--)-----><

SHA1 algorithm. Not used in *mod_auth(nz)_ibmdb2*. The `apr_sha1` is compatible to the Apache function that is used in the `htpasswd` utility.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 120 bytes.

The result of the function is VARCHAR(33). The result can be null; if the argument is null, the result is the null value.

Examples:

1)

```
INSERT INTO USERS (username, password)
VALUES ('test', apr_sha1('testpwd'))
```

2)

```
SELECT apr_sha1( 'testpwd' ) FROM SYSIBM.SYSDUMMY1
```

```
1
```

```
-----
{SHA}m08HW0aqxvmp4R11SMgZC3LJWB0=
```

```
1 record(s) selected.
```

C Stored Procedure Support

Stored procedures can minimize the network traffic and regarding the authentication module they can ease the administration. The module supports two types of stored procedures: one for user authentication and one for group authentication.

For the following 2 sections we use these 3 tables:

```
CREATE TABLE WEB.USERS (  
    USERNAME VARCHAR(40) NOT NULL,  
    PASSWORD VARCHAR(40) );
```

```
ALTER TABLE WEB.USERS  
    ADD PRIMARY KEY (USERNAME);
```

```
CREATE TABLE WEB.GROUPS (  
    GROUPNAME VARCHAR(40) NOT NULL,  
    ACTIVE     INTEGER     NOT NULL );
```

```
ALTER TABLE WEB.GROUPS  
    ADD PRIMARY KEY (GROUPNAME);
```

```
CREATE TABLE WEB.MAPPING (  
    USERNAME  VARCHAR(40) NOT NULL,  
    GROUPNAME VARCHAR(40) NOT NULL );
```

```
ALTER TABLE WEB.MAPPING  
    ADD PRIMARY KEY (USERNAME, GROUPNAME)  
    ADD FOREIGN KEY (USERNAME) REFERENCES WEB.USERS (USERNAME)  
    ADD FOREIGN KEY (GROUPNAME) REFERENCES WEB.GROUPS (GROUPNAME);
```

C.1 user authentication

The stored procedure for user authentication is responsible for returning the password of the user in question to the module. It must return exact one value - the password. If `AuthIBMDB2NoPasswd` is `On`, then the username has to be returned instead of the password.

The stored procedure must have the following parameter format:

```
CREATE PROCEDURE user_procedure_name ( IN VARCHAR, OUT VARCHAR )
```

Example:

```
CREATE PROCEDURE user_sp
(IN v_username VARCHAR(40), OUT v_password VARCHAR(40))
LANGUAGE SQL
BEGIN
    SELECT password INTO v_password FROM web.users
    WHERE username = v_username;
END@
```

If `AuthIBMDB2NoPasswd` is `On`, then the stored procedure would have to look like this:

```
CREATE PROCEDURE user_sp
(IN v_username VARCHAR(40), OUT v_password VARCHAR(40))
LANGUAGE SQL
BEGIN
    SELECT username INTO v_password FROM web.users
    WHERE username = v_username;
END@
```

C.2 group authentication

The stored procedure for group authentication is responsible for returning the groups the user in question belongs to. It must return an open cursor to the resultset.

The stored procedure must have the following parameter format:

```
CREATE PROCEDURE group_procedure_name ( IN VARCHAR )
```

Example

```
CREATE PROCEDURE group_sp
(IN v_username VARCHAR(40))
LANGUAGE SQL
DYNAMIC RESULT SETS 1
BEGIN
    DECLARE res CURSOR WITH RETURN FOR
    SELECT m.groupname FROM web.groups g, web.mapping m
    WHERE m.groupname = g.groupname AND
          m.username = v_username AND
          g.active = 1;

    OPEN res;
END@
```