Documentation for db2-hash-routines

Helmut K. C. Tessarek

5th May, 2017

db2-hash-routines is a package which provides User Defined Functions and Stored Procedures for IBM® DB2® to generate and validate hashes.

http://tessus.github.io/db2-hash-routines

Date: 2017-05-05 01:42:59 -0400 Id: a4050e7

Contents

1. db2-hash-routines

1.1. Building the library and registering the UDFs and SPs

Login as the instance user and run the script

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

The `makertn` script detects the DB2 instance directory and locates `apr-1-config` and `apu-1-config` automatically. If for some reason the script cannot set either one of the necessary variables, they have to be set manually. Uncomment and change the following variables in the `makertn` script.

```
DB2PATH=
APRPATH=
APUPATH=
```

Set `DB2PATH` to the directory where DB2 is accessed. This is usually the instance home directory.
Set APRPATH to where apr-1-config is located.
Set APUPATH to where apu-1-config is located.

The UDFs and SPs are written in ANSI C and should compile on all platforms.

The only requirements are APR and APR-util. You can get APR and APR-util at `http://apr.apache.org/`

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

```
db2 -td@ -f register.ddl
```

1.2. Description of the UDFs and SPs

This library delivers the following routines[1]:

```
bcrypt
sha256_hex
sha256
sha512
php_md5
apr_md5
apr_crypt
apr_sha1
apr_sha256
validate_pw
```

The `php_md5` routine is compatible to the PHP md5 function.
The `sha256_hex` routine returns a sha256 64-character hexadecimal hash.
The `apr_md5`, `apr_crypt`, `apr_sha1` and `bcrypt` routines are compatible to the functions used in Apache's `htpasswd` utility.
The `apr_sha256` routine returns the identifier {`SHA256`} plus the base64 encoded sha256 hash.
The `sha256` and `sha512` functions return glib2's crypt hashes (if supported).

`validate_pw` can be used to validate a password against a hash.

On systems with glibc2, the `validate_pw` routine will also validate hashes of the form `$id$salt$encrypted`. The following values of `id` are supported:

ID	Method
1	MD5
2a	Blowfish (not in mainline glibc; added in some Linux distributions)
5	SHA-256 (since glibc 2.7)
6	SHA-512 (since glibc 2.7)

Note: In win32 environments `apr_crypt` returns the output of `bcrypt`, if available. If `bcrypt` is not available, the output of `apr_md5` is returned.

[1] see Appendix A for a reference of the UDFs and SPs

A. UDF and SP reference

A.1. bcrypt

```
>>-BCRYPT--(--expression--)----------------------------------><

>>-BCRYPT--(--expression--,--hash--)--------------------------><
```

bcrypt algorithm. The `bcrypt` routine is compatible to the function used in Apache's `htpasswd` utility.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(60). 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', bcrypt('testpwd'))

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

    1
    ------------------------------------------------------------
    $2y$05$2jb66aPElSkNLT1t8e6dQepuCY2BP3JnYUh0xeV9r1PEoOGyOLkym

      1 record(s) selected.

3)
    CALL bcrypt('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : $2y$05$WYSu1X6PVAORa.aPSjrdv.S6hOp.AYSnNRT521rmLRjD4Mj9UY6ve

      Return Status = 0
```

A.2. sha256_hex

```
>>-SHA256_HEX--(--expression--)--------------------------------><

>>-SHA256_HEX--(--expression--,--hash--)----------------------><
```

SHA256 algorithm. The `sha256_hex` routine returns a 64-character hexadecimal hash.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(64). 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', sha256_hex('testpwd'))

2)
    SELECT sha256_hex('testpwd') FROM SYSIBM.SYSDUMMY1

    1
    ----------------------------------------------------------------
    a85b6a20813c31a8b1b3f3618da796271c9aa293b3f809873053b21aec501087

      1 record(s) selected.

3)
    CALL sha256_hex('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : a85b6a20813c31a8b1b3f3618da796271c9aa293b3f809873053b21
    aec501087

      Return Status = 0
```

A.3. sha256

```
>>-SHA256--(--expression--+---------+--)----------------------><
                          '-,--salt-'

>>-SHA256--(--expression--+---------+--,--hash--)--------------><
                          '-,--salt-'
```

SHA256 algorithm. The `sha256` routine returns a glibc2's crypt hash. If the system's crypt does not support sha-256, an `SQLSTATE 39702` is returned.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.
An optional salt can be specified, which must be a eight-character string chosen from the set [a-z-Z0-9./]. If the salt is not exactly eight characters long, an `SQLSTATE 39703` is returned. If the salt contains invalid characters, an `SQLSTATE 39704` is returned.

The result of the function is CHAR(55). The result can be null; if one of the arguments is null, the result is the null value.

Examples:

```
1)
    INSERT INTO USERS (username, password)
       VALUES ('test', sha256('testpwd'))

2)
    SELECT sha256( 'testpwd' ) FROM SYSIBM.SYSDUMMY1

    1
    -------------------------------------------------------
    $5$S.LqPR7Z$273zPncMdmJ0dE1WdLldWVBmaHSDUDl8/tW8At8Hc0A

      1 record(s) selected.

3)
    CALL sha256('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : $5$vSDCZr2d$rfh.aDopE5l3lm26AwwcIYnuVdV7/9QBACWukqYyV3/
```

```
      Return Status = 0

4)
    SELECT sha256('testpwd', '12345678') FROM SYSIBM.SYSDUMMY1

    1
    ------------------------------------------------------
    $5$12345678$.oVAnOr/.FK8fYNiFPvoXPQvEOT9Calecygw6K9wIb9

      1 record(s) selected.

5)
    CALL sha256('testpwd', '12345678', ?)

    Value of output parameters
    --------------------------
    Parameter Name  : HASH
    Parameter Value : $5$12345678$.oVAnOr/.FK8fYNiFPvoXPQvEOT9Calecygw6K9wIb9

    Return Status = 0
```

A.4. sha512

```
>>-SHA512--(--expression--+---------+--)----------------------><
                         '-,--salt-'

>>-SHA512--(--expression--+---------+--,--hash--)--------------><
                         '-,--salt-'
```

SHA512 algorithm. The `sha512` routine returns a glibc2's crypt hash. If the system's crypt does not support sha-512, an `SQLSTATE 39702` is returned.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.
An optional salt can be specified, which must be a eight-character string chosen from the set [a-z-Z0-9./]. If the salt is not exactly eight characters long, an `SQLSTATE 39703` is returned. If the salt contains invalid characters, an `SQLSTATE 39704` is returned.

The result of the function is CHAR(98). The result can be null; if one of the arguments is null, the result is the null value.

Examples:

```
1)
    INSERT INTO USERS (username, password)
       VALUES ('test', sha512('testpwd'))

2)
    SELECT sha512( 'testpwd' ) FROM SYSIBM.SYSDUMMY1

    1
    ------------------------------------------------------------------------
    ----------------------
    $6$cD33haq7$dl.RqEaLamlesTPVzSIQr4N1MY3BsVZ76VS8qNte0IOIWO2XorMg8U797KKOFGm
    X8dJhT3WuF6p17HmvvoQ6Q/

      1 record(s) selected.

3)
    CALL sha512('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
```

```
  Parameter Value : $6$1W.m9JN1$Dh.VPl7vy.igGaeDUdDWw6ZlDOxufwDWmOukpOYknPt
djxiSM2yzWBkzHffalb/2axNHPqEi9UUzXUbSm4LGa/

  Return Status = 0

4)
  SELECT sha512('testpwd', '12345678') FROM SYSIBM.SYSDUMMY1

  1
  ------------------------------------------------------------------------
  ----------------------
  $6$12345678$tlHrypdWTz6FqubBpgL/ePlxr4lZuQ8OK1zfV6zWUmGJSz.5kGWwQGjg69Qm1Bm
  3.DvILruqA61o3EHsxSoko1

  1 record(s) selected.

5)
  CALL sha512('testpwd', '12345678', ?)

  Value of output parameters
  --------------------------
  Parameter Name  : HASH
  Parameter Value : $6$12345678$tlHrypdWTz6FqubBpgL/ePlxr4lZuQ8OK1zfV6zWUmGJS
z.5kGWwQGjg69Qm1Bm3.DvILruqA61o3EHsxSoko1

  Return Status = 0
```

A.5. php_md5

```
>>-PHP_MD5--(--expression--)-----------------------------------><

>>-PHP_MD5--(--expression--,--hash--)-------------------------><
```

MD5 hash. The `php_md5` routine is compatible to the PHP md5 function.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(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', php_md5('testpwd'))

2)
    SELECT php_md5( 'testpwd' ) FROM SYSIBM.SYSDUMMY1

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

      1 record(s) selected.

3)
    CALL php_md5('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : 342df5b036b2f28184536820af6d1caf

      Return Status = 0
```

A.6. apr_md5

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

>>-APR_MD5--(--expression--,--hash--)-------------------------><
```

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

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(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
    ------------------------------------
    $apr1$GfVmOTyJ$n7F1Vkwl/kX8MLgTJq1lp1

      1 record(s) selected.

3)
    CALL apr_md5('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : $apr1$GfVmOTyJ$n7F1Vkwl/kX8MLgTJq1lp1

      Return Status = 0
```

A.7. apr_crypt

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

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

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

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(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
    -------------
    cqs7uOvz8KBlk

      1 record(s) selected.

3)
    CALL apr_crypt('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : cqs7uOvz8KBlk

      Return Status = 0
```

A.8. apr_sha1

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

>>-APR_SHA1--(--expression--,--hash--)------------------------><
```

SHA1 algorithm. The `apr_sha1` routine is compatible to the function used in Apache's `htpasswd` utility.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(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}mO8HWOaqxvmp4Rl1SMgZC3LJWBO=

      1 record(s) selected.

3)
    CALL apr_sha1('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : {SHA}mO8HWOaqxvmp4Rl1SMgZC3LJWBO=

      Return Status = 0
```

A.9. apr_sha256

```
>>-APR_SHA256--(--expression--)-------------------------------><

>>-APR_SHA256--(--expression--,--hash--)----------------------><
```

SHA256 algorithm. The `apr_sha256` routine returns the identifier {`SHA256`} plus the base64 encoded sha256 hash.

The argument can be a character string that is either a CHAR or VARCHAR not exceeding 4096 bytes.

The result of the function is CHAR(52). 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_sha256('testpwd'))

2)
    SELECT apr_sha256( 'testpwd' ) FROM SYSIBM.SYSDUMMY1

    1
    ----------------------------------------------------
    {SHA256}qFtqIIE8Maixs/NhjaeWJxyaopOz+AmHMFOyGuxQEIc=

      1 record(s) selected.

3)
    CALL apr_sha256('testpwd', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : HASH
      Parameter Value : {SHA256}qFtqIIE8Maixs/NhjaeWJxyaopOz+AmHMFOyGuxQEIc=

      Return Status = 0
```

A.10. validate_pw

```
>>-VALIDATE_PW--(--password--,--hash--)-----------------------><

>>-VALIDATE_PW--(--password--,--hash--,--is_valid--)-----------><
```

This routine can be used to validate a password against a hash.

The two input arguments can be character strings that are either a CHAR or VARCHAR not exceeding 4096 bytes (password) and 120 bytes (hash). The second parameter (hash) must not be empty, otherwise an `SQLSTATE 39701` is returned.

The result of the routine is an INTEGER. If the password is valid, 1 is returned. If the password is not valid, 0 is returned. The result can be null; if the argument is null, the result is the null value.

Examples:

```
1)
    SELECT validate_pw('testpwd', 'cqs7uOvz8KBlk') FROM SYSIBM.SYSDUMMY1"

    1
    -----------
              1

      1 record(s) selected.

2)
    CALL validate_pw('testpwd', 'cqs7uOvz8KBlk', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : IS_VALID
      Parameter Value : 1

      Return Status = 0

3)
    CALL validate_pw('testpwd', '0123456789abcdef', ?)

      Value of output parameters
      --------------------------
      Parameter Name  : IS_VALID
```

```
Parameter Value : 0

Return Status = 0
```

Date: 2017-05-05 01:42:59 -0400 Id: a4050e76a5eb2923533c36f0bfb1199806ced229