HASH Message Digest Framework

Introduction

Message Digest (hash) engine. Allows direct or incremental processing of arbitrary length messages using a variety of hashing algorithms.

Installing/Configuring

Requirements

The Hash extension requires no external libraries and is enabled by default as of PHP 5.1.2. It may be explicitly disabled by using the --disable-hash switch to configure. Earlier versions of PHP may incorporate the Hash extension by installing the » PECL module.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension defines a Hashing Context resource returned by hash_init().

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

HASH_HMAC (integer)
Optional flag for hash_init(). Indicates that the HMAC digest-keying algorithm should be applied to the current hashing context.

Hash Functions

hash_algos

hash_algos -- Return a list of registered hashing algorithms

Description

array **hash_algos** (void)

Return Values

Returns a numerically indexed array containing the list of supported hashing algorithms.

Examples

Example #1 - hash_algos() example

As of PHP 5.1.2, hash_algos() will return the following list of algorithm names.

```
<?php
print_r(hash_algos());
?>
```

The above example will output:

```
Array
(
    [0] => md4
    [1] => md5
    [2] => sha1
    [3] => sha256
    [4] => sha384
    [5] => sha512
    [6] => ripemd128
    [7] => ripemd160
    [8] => whirlpool
    [9] => tiger128,3
   [10] => tiger160,3
   [11] => tiger192,3
   [12] => tiger128,4
   [13] => tiger160,4
   [14] => tiger192,4
   [15] => snefru
   [16] => gost
   [17] => adler32
   [18] => crc32
   [19] => crc32b
   [20] => haval128,3
   [21] => haval160,3
   [22] => haval192,3
   [23] => haval224,3
   [24] => haval256,3
```

```
    [25] => haval128,4
    [26] => haval160,4
    [27] => haval192,4
    [28] => haval224,4
    [29] => haval256,4
    [30] => haval128,5
    [31] => haval160,5
    [32] => haval192,5
    [33] => haval224,5
    [34] => haval256,5
)
```

hash_copy

hash_copy -- Copy hashing context

Description

resource **hash_copy** (resource `$context`)

Parameters

`context`
Hashing context returned by hash_init().

Return Values

Returns a copy of Hashing Context resource.

Examples

Example #2 - hash_copy() example

```
<?php
$context = hash_init("md5");
hash_update($context, "data");

/* copy context to be able to continue using it */
$copy_context = hash_copy($context);

echo hash_final($context), "\n";

hash_update($copy_context, "data");
echo hash_final($copy_context), "\n";
?>
```

The above example will output:

```
8d777f385d3dfec8815d20f7496026dc
511ae0b1c13f95e5f08f1a0dd3da3d93
```

hash_file

hash_file -- Generate a hash value using the contents of a given file

Description

string **hash_file** (string `$algo`, string `$filename` [, bool `$raw_output`])

Parameters

`algo`
: Name of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..)

`filename`
: URL describing location of file to be hashed; Supports fopen wrappers.

`raw_output`
: When set to **TRUE**, outputs raw binary data. Default value (**FALSE**) outputs lowercase hexits.

Return Values

Returns a string containing the calculated message digest as lowercase hexits unless *`raw_output`* is set to true in which case the raw binary representation of the message digest is returned.

Examples

Example #3 - Using hash_file()

```
<?php
/* Create a file to calculate hash of */
file_put_contents('example.txt', 'The quick brown fox jumped over the lazy
dog.');

echo hash_file('md5', 'example.txt');
?>
```

The above example will output:

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

See Also

- hash()
- hash_hmac_file()
- hash_update_file()

hash_final

hash_final -- Finalize an incremental hash and return resulting digest

Description

string **hash_final** (resource `$context` [, bool `$raw_output`])

Parameters

`context`
Hashing context returned by hash_init().

`raw_output`
When set to **TRUE**, outputs raw binary data. Default value (**FALSE**) outputs lowercase hexits.

Return Values

Returns a string containing the calculated message digest as lowercase hexits unless *`raw_output`* is set to true in which case the raw binary representation of the message digest is returned.

Examples

Example #4 - hash_final() example

```
<?php
$ctx = hash_init('sha1');
hash_update($ctx, 'The quick brown fox jumped over the lazy dog.');
echo hash_final($ctx);
?>
```

The above example will output:

```
c0854fb9fb03c41cce3802cb0d220529e6eef94e
```

See Also

- hash_init()
- hash_update()

- hash_update_stream()
- hash_update_file()

hash_hmac_file

hash_hmac_file -- Generate a keyed hash value using the HMAC method and the contents of a given file

Description

string **hash_hmac_file** (string `$algo`, string `$filename`, string `$key` [, bool `$raw_output`])

Parameters

`algo`
Name of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..)

`filename`
URL describing location of file to be hashed; Supports fopen wrappers.

`key`
Shared secret key used for generating the HMAC variant of the message digest.

`raw_output`
When set to **TRUE**, outputs raw binary data. Default value (**FALSE**) outputs lowercase hexits.

Return Values

Returns a string containing the calculated message digest as lowercase hexits unless *`raw_output`* is set to true in which case the raw binary representation of the message digest is returned.

Examples

Example #5 - hash_hmac_file() **example**

```
<?php
/* Create a file to calculate hash of */
file_put_contents('example.txt', 'The quick brown fox jumped over the lazy
dog.');

echo hash_hmac_file('md5', 'example.txt', 'secret');
?>
```

The above example will output:

```
7eb2b5c37443418fc77c136dd20e859c
```

See Also

- hash_hmac()
- hash_file()

hash_hmac

hash_hmac -- Generate a keyed hash value using the HMAC method

Description

string **hash_hmac** (string `$algo`, string `$data`, string `$key` [, bool `$raw_output`])

Parameters

`algo`
: Name of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..)

`data`
: Message to be hashed.

`key`
: Shared secret key used for generating the HMAC variant of the message digest.

`raw_output`
: When set to **TRUE**, outputs raw binary data. Default value (**FALSE**) outputs lowercase hexits.

Return Values

Returns a string containing the calculated message digest as lowercase hexits unless *`raw_output`* is set to true in which case the raw binary representation of the message digest is returned.

Examples

Example #6 - hash_hmac() **example**

```
<?php
echo hash_hmac('ripemd160', 'The quick brown fox jumped over the lazy dog.',
'secret');
?>
```

The above example will output:

```
b8e7ae12510bdfb1812e463a7f086122cf37e4f7
```

See Also

- hash()
- hash_init()
- hash_hmac_file()

hash_init

hash_init -- Initialize an incremental hashing context

Description

resource **hash_init** (string `$algo` [, int `$options` [, string `$key`]])

Parameters

`algo`
: Name of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..)

`options`
: Optional settings for hash generation, currently supports only one option: **HASH_HMAC**. When specified, the *`key`* *must* be specified.

`key`
: When **HASH_HMAC** is specified for *`options`*, a shared secret key to be used with the HMAC hashing method must be supplied in this parameter.

Return Values

Returns a Hashing Context resource for use with hash_update(), hash_update_stream(), hash_update_file(), and hash_final().

Examples

Example #7 - Incremental hashing example

```
<?php
$ctx = hash_init('md5');
hash_update($ctx, 'The quick brown fox ');
hash_update($ctx, 'jumped over the lazy dog.');
echo hash_final($ctx);
?>
```

The above example will output:

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

See Also

- hash()
- hash_file()
- hash_hmac()
- hash_hmac_file()

hash_update_file

hash_update_file -- Pump data into an active hashing context from a file

Description

bool **hash_update_file** (resource `$context`, string `$filename` [, resource `$context`])

Parameters

`context`
: Hashing context returned by hash_init().

`filename`
: URL describing location of file to be hashed; Supports fopen wrappers.

`context`
: Stream context as returned by stream_context_create().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- hash_init()
- hash_update()
- hash_update_stream()
- hash_final()
- hash()
- hash_file()

hash_update_stream

hash_update_stream -- Pump data into an active hashing context from an open stream

Description

int **hash_update_stream** (resource `$context`, resource `$handle` [, int `$length`])

Parameters

`context`
Hashing context returned by hash_init().

`handle`
Open file handle as returned by any stream creation function.

`length`
Maximum number of characters to copy from *`handle`* into the hashing context.

Return Values

Actual number of bytes added to the hashing context from *`handle`*.

Examples

Example #8 - hash_update_stream() example

```
<?php
$fp = tmpfile();
fwrite($fp, 'The quick brown fox jumped over the lazy dog.');
rewind($fp);

$ctx = hash_init('md5');
hash_update_stream($ctx, $fp);
echo hash_final($ctx);
?>
```

The above example will output:

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

See Also

- hash_init()
- hash_update()
- hash_final()
- hash()
- hash_file()

hash_update

hash_update -- Pump data into an active hashing context

Description

bool **hash_update** (resource `$context`, string `$data`)

Parameters

`context`
: Hashing context returned by hash_init().

`data`
: Message to be included in the hash digest.

Return Values

Returns **TRUE**.

See Also

- hash_init()
- hash_update_file()
- hash_update_stream()
- hash_final()

hash

hash -- Generate a hash value (message digest)

Description

string **hash** (string `$algo`, string `$data` [, bool `$raw_output`])

Parameters

`algo`
: Name of selected hashing algorithm (i.e. "md5", "sha256", "haval160,4", etc..)

`data`
: Message to be hashed.

`raw_output`
: When set to **TRUE**, outputs raw binary data. Default value (**FALSE**) outputs lowercase hexits.

Return Values

Returns a string containing the calculated message digest as lowercase hexits unless *`raw_output`* is set to true in which case the raw binary representation of the message digest is returned.

Examples

Example #9 - A hash() example

```
<?php
echo hash('ripemd160', 'The quick brown fox jumped over the lazy dog.');
?>
```

The above example will output:

```
ec457d0a974c48d5685a7efa03d137dc8bbde7e3
```

See Also

- hash_file()

- hash_hmac()
- hash_init()