session-regenerate-id

session_regenerate_id

Update the current session id with a newly generated one

ENTITY

bool

session_regenerate_id

bool

delete_old_session

ENTITY

`session_regenerate_id` will replace the current session id with a new one, and keep the current session information.

When session.use_trans_sid is enabled, output must be started after `session_regenerate_id` call. Otherwise, old session ID is used.

> Currently, session_regenerate_id does not handle an unstable network well, e.g. Mobile and WiFi network. Therefore, you may experience a lost session by calling session_regenerate_id.
>
> You should not destroy old session data immediately, but should use destroy time-stamp and control access to old session ID. Otherwise, concurrent access to page may result in inconsistent state, or you may have lost session, or it may cause client (browser) side race condition and may create many session ID needlessly. Immediate session data deletion disables session hijack attack detection and prevention also.

ENTITY

`delete_old_session`
Whether to delete the old associated session file or not. You should not delete old session if you

need to avoid races caused by deletion or detect/avoid session hijack attacks.

ENTITY

ENTITY

ENTITY

A `session_regenerate_id` example

```
<?php
// NOTE: This code is not fully working code, but an example!

session_start();

// Check destroyed time-stamp
if (isset($_SESSION['destroyed'])
    && $_SESSION['destroyed'] < time() - 300) {
    // Should not happen usually. This could be attack or due to unstable network.
    // Remove all authentication status of this users session.
    remove_all_authentication_flag_from_active_sessions($_SESSION['userid']);
    throw(new DestroyedSessionAccessException);
}

$old_sessionid = session_id();

// Set destroyed timestamp
$_SESSION['destroyed'] = time(); // session_regenerate_id() saves old session data

// Simply calling session_regenerate_id() may result in lost session, etc.
// See next example.
session_regenerate_id();

// New session does not need destroyed timestamp
unset($_SESSION['destroyed']);

$new_sessionid = session_id();

echo "Old Session: $old_sessionid<br />";
```

```
echo "New Session: $new_sessionid<br />";

print_r($_SESSION);
?>
```

Current session module does not handle unstable network well. You should manage session ID to avoid lost session by session_regenerate_id.

Avoiding lost session by `session_regenerate_id`

```
<?php
// NOTE: This code is not fully working code, but an example!
// my_session_start() and my_session_regenerate_id() avoid lost sessions by
// unstable network. In addition, this code may prevent exploiting stolen
// session by attackers.

function my_session_start() {
    session_start();
    if (isset($_SESSION['destroyed'])) {
       if ($_SESSION['destroyed'] < time()-300) {
           // Should not happen usually. This could be attack or due to unstable network.
           // Remove all authentication status of this users session.
           remove_all_authentication_flag_from_active_sessions($_SESSION['userid']);
           throw(new DestroyedSessionAccessException);
       }
       if (isset($_SESSION['new_session_id'])) {
           // Not fully expired yet. Could be lost cookie by unstable network.
           // Try again to set proper session ID cookie.
           // NOTE: Do not try to set session ID again if you would like to remove
           // authentication flag.
           session_commit();
           session_id($_SESSION['new_session_id']);
           // New session ID should exist
           session_start();
           return;
       }
   }
}
```

```
function my_session_regenerate_id() {
    // New session ID is required to set proper session ID
    // when session ID is not set due to unstable network.
    $new_session_id = session_create_id();
    $_SESSION['new_session_id'] = $new_session_id;

    // Set destroy timestamp
    $_SESSION['destroyed'] = time();

    // Write and close current session;
    session_commit();

    // Start session with new session ID
    session_id($new_session_id);
    ini_set('session.use_strict_mode', 0);
    session_start();
    ini_set('session.use_strict_mode', 1);

    // New session does not need them
    unset($_SESSION['destroyed']);
    unset($_SESSION['new_session_id']);
}
?>
```

ENTITY

`session_id`, `session_create_id`, `session_start`, `session_destroy`, `session_reset`, `session_name`

□□□ #2
□ □□□ □□□ 2026-10-31 18:01:02 UTC
□ □□□ □□□ 2026-10-31 18:01:03 UTC