
Get in, hurry up!
The mysqlnd plugin talk starts!

Ulf Wendel, MySQL/Sun/Oracle/WhatIsNext

The mysqlnd green house

Gardening mysqlnd – Concepts and Internals

Use the URL not the slides!

The slides are a quick copy of the real stuff at:

http://blog.ulf-wendel.de/mysqlnd_plugin_ipc2010.html

Target audience

PHP developers with
C programming knowledge.

Familarity with PHP
extension writing is benefitial.

„Das MySQL-Treibhaus erweitern ...
Level: Experte“, Session description

Quick poll: are you a man?

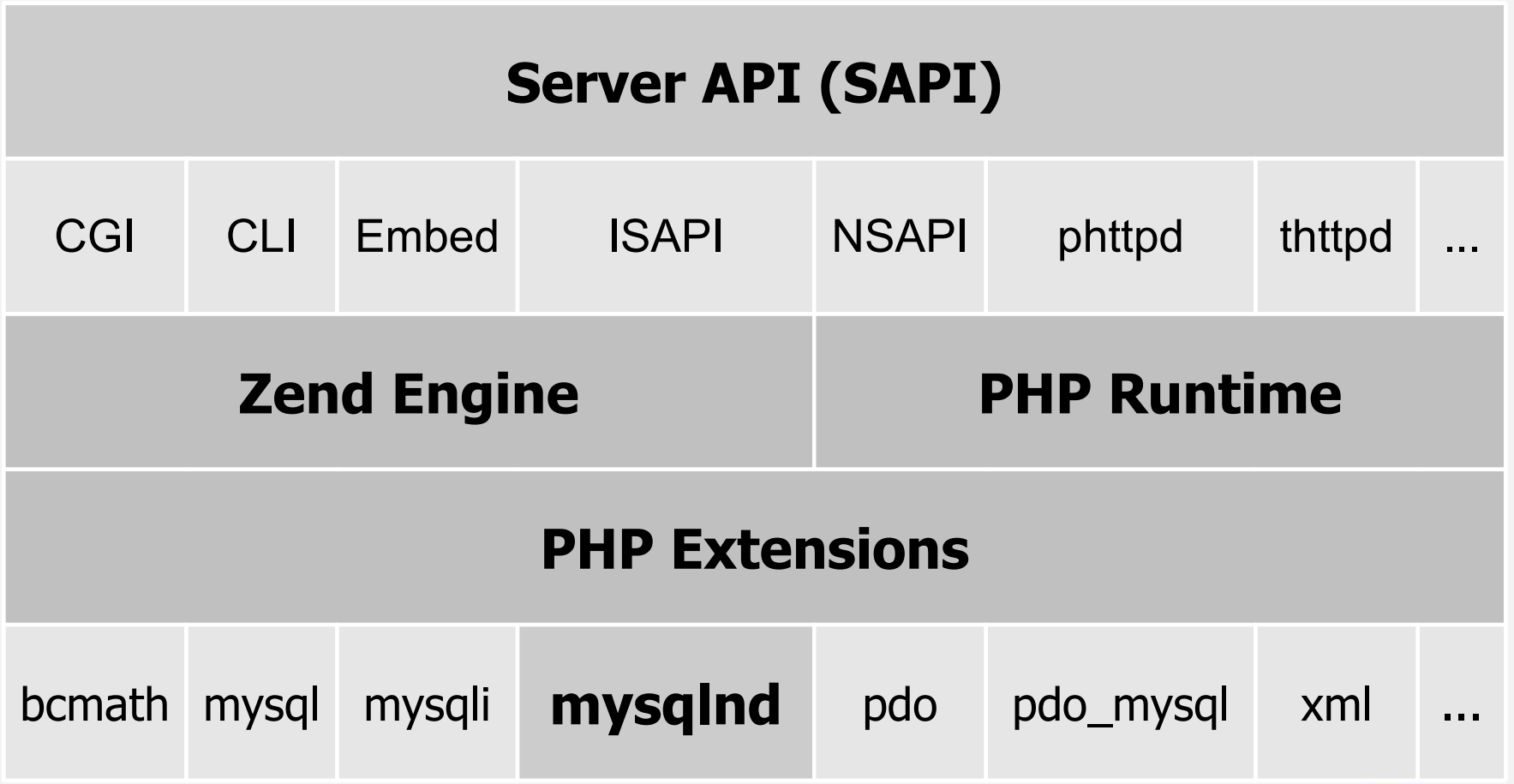
Raise your hand, if you call yourself a man.
Not a lamer, not a sissy - a real man.

Cowboys, who of you calls himself an Open Source developer?

Please state your name!

The pub, the idea, the demo

The MySQL native driver for PHP



Replacement for libmysql

```
$mysqli = new mysqli(...);
$mysql = mysql_connect(...);
$pdo = new PDO(...);
```


PHP (SAPI, Zend, Runtime)

PHP Extensions

mysqli	mysql	PDO_MYSQL

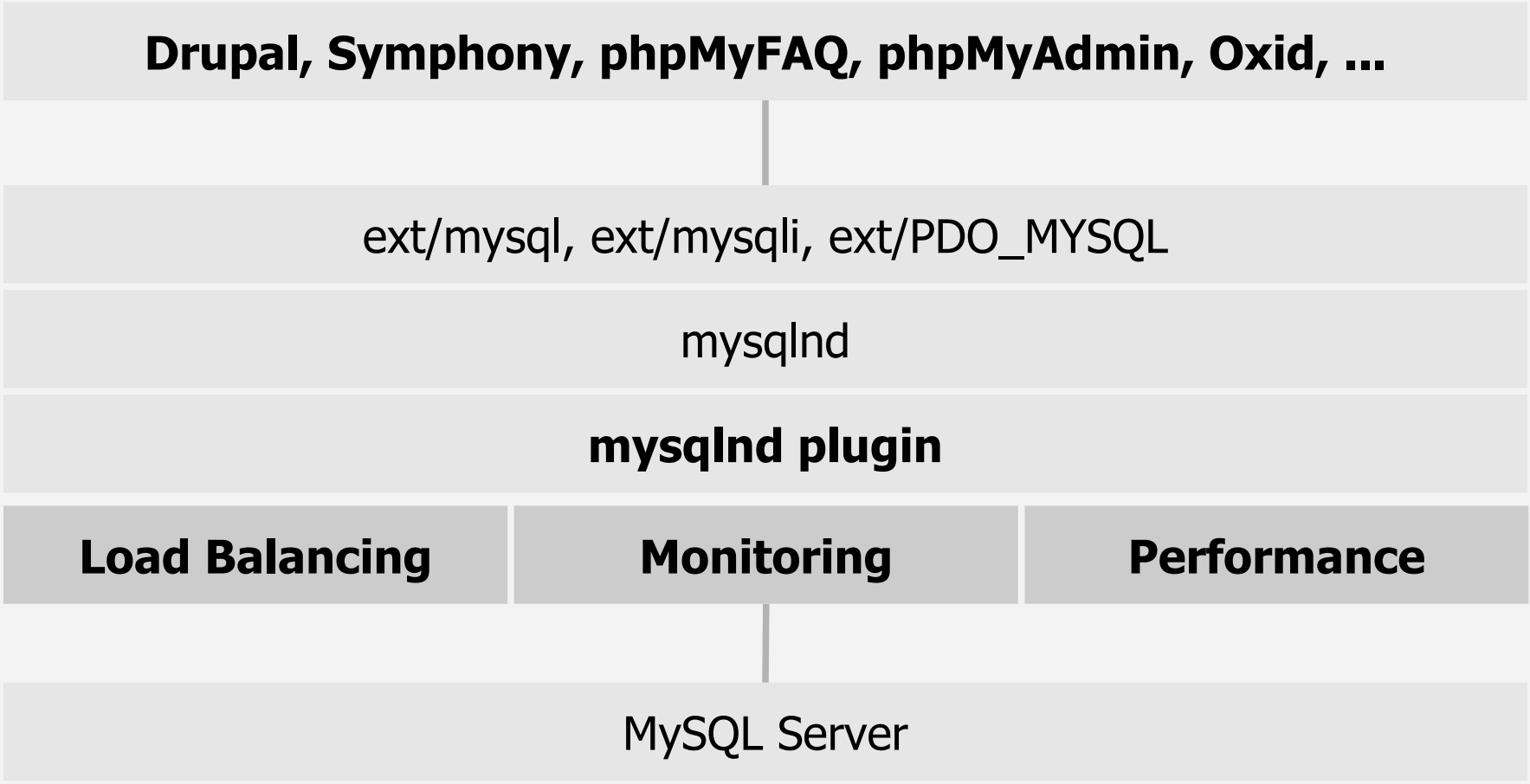
mysqlnd (or MySQL Client Library AKA libmysql AKA Connector/C)

MySQL Server

How to grow/extend mysqlnd

*.php	PHP Application
*.c	ext/mysql, ext/mysqli, ext/PDO_MYSQL
*.c	mysqlnd
*.c	**mysqlnd plugin**
*.c	MySQL Server

What mysqlnd plugins can do!



What are mysqlnd plugins?

- Extension
  - Adds new functionality
- Proxy
  - Surrogate
  - Intermediary

Plugin vs. MySQL Proxy (I)

Hardware	Software		
Application server	PHP application	C/Java/.NET/PHP... application	
	mysqlnd plugin	**MySQL Proxy**	
Dedicated server			**MySQL Proxy**
Database server	MySQL Server		

Choose: C API or wire protocol

Layer	Software	
	PHP application	C/Java/.NET/PHP... application
C API	**mysqlnd plugin**	
Wire protocol	**mysqlnd plugin**	**MySQL Proxy**

Follow me or leave the room!

Start your GCC's, boys!

- You want mysqlnd plugins because
  - 100% transparent = 100% compatible
  - Cure applications without touching .php
  - No extra software, no MySQL Proxy!
  - Extend, add driver functionality
- All you need to is
  - … write comments into comment files (*.c)

mysqlnd modules

PHP Extension Infrastructure			
php_mysqlnd.c			
Core			
mysqlnd.c			
Modules			**Statistics**
Connection	**Resultset**	**Resultset Meta**	*statistics.c
mysqlnd.c	*_result.c	*_result_meta.c	
Statement	**Network**	**Wire protocol**	
*_ps.c	*_net.c	*_wireprotocol.c	

mysqlnd modules are objects

```
struct st_mysqlnd_conn_methods {
  void (*init)(MYSQLND * conn TSRMLS_DC);
  enum_func_status (*query)(
     MYSQLND *conn, const char *query,
     unsigned int query_len TSRMLS_DC);
  /* ... 50+ methods not shown */
};

struct st_mysqlnd_connection {
  /* properties */
  char       *host;
  unsigned int  host_len;
  /* ... ... */
  /* methods */
  struct st_mysqlnd_conn_methods *m;
};
```

The classes

	# public	#private (not final!)	#total
Connection	48	5	53
Resultset	26	0	26
Resultset Meta	6	0	6
Statement	35	1	35
Network	11	0	11
Wire protocol	10	0	10
Total	**136**	**6**	**142**

Revision 299098 = PHP 5.3 on May, 7 2010 -
Andrey continued working since then...

Extending Connection: methods

```
/* a place to store orginal function table */
struct st_mysqlnd_conn_methods org_methods;

void minit_register_hooks(TSRMLS_D) {
  /* active function table */
  struct st_mysqlnd_conn_methods * current_methods
   = mysqlnd_conn_get_methods();

  /* backup original function table */
  memcpy(&org_methods, current_methods,
    sizeof(struct st_mysqlnd_conn_methods);

  /* install new methods */
  current_methods->query =
      MYSQLND_METHOD(my_conn_class, query);
}
```

Extending: parent methods

```
MYSQLND_METHOD(my_conn_class, query)(MYSQLND *conn,
  const char *query, unsigned int query_len TSRMLS_DC) {

    php_printf("my_conn_class::query(query = %s)\n",
       query);

    query = "SELECT 'query rewritten' FROM DUAL";
    query_len = strlen(query);

    return org_methods.query(conn, query, query_len);
  }
}
```

Extending: properties (concept)

OO concept	mysqlnd C struct member	comment
Methods	struct object_methods * methods	Function table
Properties	c_type member	Parent properties
Extended properties	void ** plugin_data	List of void*. One void* per registered plugin

Extending: properties (API)

```
void minit_register_hooks(TSRMLS_D) {
  /* obtain unique plugin ID */
  my_plugin_id = mysqlnd_plugin_register();
  /* snip - see Extending Connection: methods */
}

static PROPS** get_conn_properties(const MYSQLND *conn TSRMLS_DC) {

  PROPS** props;
  props = (PROPS**)mysqlnd_plugin_get_plugin_connection_data
                      (conn, my_plugin_id);

  if (!props || !(*props)) {
    *props = mnd_pecalloc
                (1, sizeof(MY_CONN_PROPERTIES), conn->persistent);
    (*props)->query_counter = 0;
  }
  return props;
}
```

Daddy, it is a plugin API !

- mysqlnd_plugin_register()
- mysqlnd_plugin_count()

- mysqlnd_plugin_get_plugin_connection_data()
- mysqlnd_plugin_get_plugin_[result|stmt]_data()
- mysqlnd_plugin_get_plugin_[net|protocol]_data()

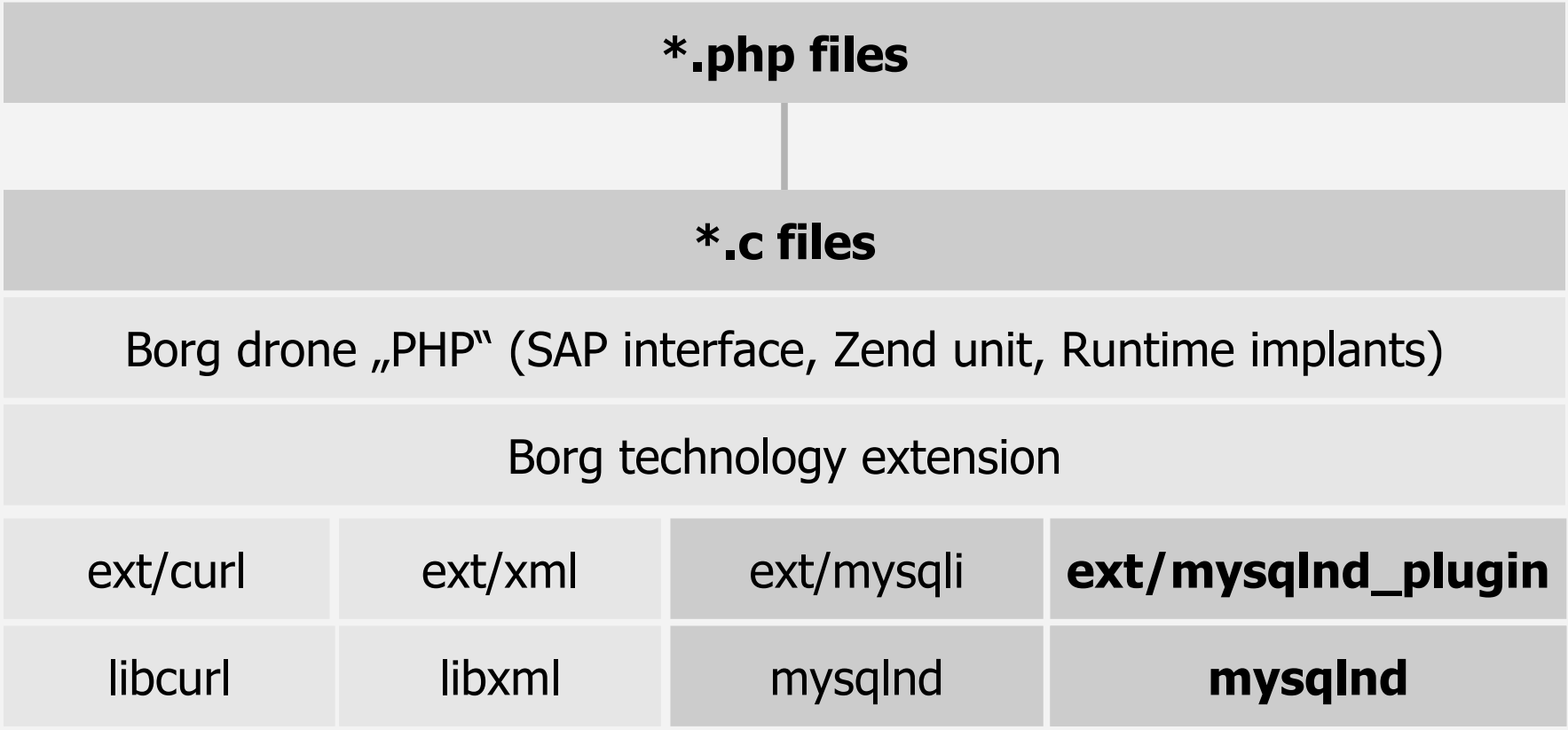
- mysqlnd_conn_get_methods()
- mysqlnd_[result|result_meta]_get_methods()
- mysqlnd_[stmt|net|protocol]_get_methods()

Risks: a (silly) man's world

- Security: sissy
- Limitations: use your leadfoot
- Chaining: alphabetical = random order

- Recommended to call parent methods
- Recommended to be cooperative

On PHP, the borg and ladies



What borg technology can do!

```
class proxy extends mysqlnd_plugin_connection {
  public function connect($host, ...) {
    /* security */
    $host = '127.0.0.1';
    return parent::connect($host);
  }
  public function query($query, ...) {
    error_log($query);
    return parent::query($query);
  }
}
mysqlnd_plugin_set_conn_proxy(new proxy());
(auto_prepend.inc.php)

any_php_mysql_app_main();
(index.php)
```

Userspace plugin motivation

- Availability: maximum
- Creative potential: endless
- Ease of use: absolutely
- Fun factor: tremendous

- US citizen: you must read and comply to the following security rules

- Security, Limitations, Chaining: consult homeland security
- The C API has not been designed to be exposed to the userspace
- Continued on page PHP_INT_MAX.

Starting w. ext/mysqlnd_plugin

- How to create an extension? Choose!
  - Ueber-lady book (ULB) templates
  - Extension generators
  - Striping an existing extension
  - Extension skeleton

Repetition: MINIT of a plugin

```
static PHP_MINIT_FUNCTION(mysqlnd_plugin) {
  /* globals, ini entries, resources, classes */

  /* register mysqlnd plugin */
  mysqlnd_plugin_id = mysqlnd_plugin_register();

  conn_m = mysqlnd_get_conn_methods();
  memcpy(org_conn_m, conn_m,
    sizeof(struct st_mysqlnd_conn_methods));

  conn_m->query = MYSQLND_METHOD(mysqlnd_plugin_conn, query);
  conn_m->connect = MYSQLND_METHOD(mysqlnd_plugin_conn, connect);
}
(my_php_mysqlnd_plugin.c)

enum_func_status MYSQLND_METHOD(mysqlnd_plugin_conn, query)(/* ...
*/) {
  /* ... */
}
(my_mysqlnd_plugin.c)
```

Task analysis: C to userspace

```
class proxy extends mysqlnd_plugin_connection {
  public function connect($host, ...) { .. }
}
mysqlnd_plugin_set_conn_proxy(new proxy());
```

- write a class "mysqlnd_plugin_connection" in C (->ULB)
- accept and register proxy object through "mysqlnd_plugin_set_conn_proxy()" (->ULB)
- call userspace proxy methods from C (-> ULB or - optimization - zend_interfaces.h)

Calling userspace

```
MYSQLND_METHOD(my_conn_class,connect)(
  MYSQLND *conn, const char *host /* ... */ TSRMLS_DC) {

  enum_func_status ret = FAIL;
  zval * global_user_conn_proxy = fetch_userspace_proxy();

  if (global_user_conn_proxy) {
    /* call userspace proxy */
    ret = MY_ZEND_CALL_METHOD_WRAPPER
            (global_user_conn_proxy, host, /*...*/);
  } else {
    /* or original mysqlnd method = do nothing, be transparent */
    ret = org_methods.connect(conn, host, user, passwd,
                              passwd_len, db, db_len, port,
                              socket, mysql_flags TSRMLS_CC);
  }
  return ret;
}
(my_mysqlnd_plugin.c)
```

Simple arguments

```
MYSQLND_METHOD(my_conn_class,connect)(/* ... */, const char *host,
/* ...*/) {
  /* ... */
  if (global_user_conn_proxy) {
    /* ... */
    zval* zv_host;
    MAKE_STD_ZVAL(zv_host);
    ZVAL_STRING(zv_host, host, 1);
    MY_ZEND_CALL_METHOD_WRAPPER
      (global_user_conn_proxy, zv_retval, zv_host /*, ...*/);
    zval_ptr_dtor(&zv_host);
    /* ... */
  }
  /* ... */
}
(my_mysqlnd_plugin.c)
```

Structs as arguments

```
MYSQLND_METHOD(my_conn_class, connect)(
  MYSQLND *conn, /* ...*/) {
  /* ... */
  if (global_user_conn_proxy) {
    /* ... */
    zval* zv_conn;
    ZEND_REGISTER_RESOURCE
      (zv_conn, (void *)conn, le_mysqlnd_plugin_conn);
    MY_ZEND_CALL_METHOD_WRAPPER
      (global_user_conn_proxy, zv_retval, zv_conn,/*, ...*/);
    zval_ptr_dtor(&zv_conn);
    /* ... */
  }
  /* ... */
}
```

Are we done ?!

```
class proxy extends mysqlnd_plugin_connection {
  public function connect($conn, $host, ...) {
    /* "pre" hook */
    printf("Connecting to host = '%s'\n", $host);
    return parent::connect($conn);
  }

  public function query($conn, $query) {
    /* "post" hook */
    $ret = parent::query($conn, $query);
    printf("Query = '%s'\n", $query);
    return $ret;
  }
}
mysqlnd_plugin_set_conn_proxy(new proxy());
```

Buildin class

```
PHP_METHOD("mysqlnd_plugin_connection", connect) {
  /* ... simplified! ... */
  zval* mysqlnd_rsrc;
  MYSQLND* conn;
  char* host; int host_len;
  if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "rs",
       &mysqlnd_rsrc, &host, &host_len) == FAILURE) {
    RETURN_NULL();
  }
  ZEND_FETCH_RESOURCE(conn, MYSQLND* conn, &mysqlnd_rsrc, -1,
    "Mysqlnd Connection", le_mysqlnd_plugin_conn);
  if (PASS == org_methods.connect(conn, host, /* simplified! */
TSRMLS_CC))
    RETVAL_TRUE;
  else
    RETVAL_FALSE;
}
(my_mysqlnd_plugin_classes.c)
```

The End: implementors wanted!

Those who have stated their name at the beginning:

Would you mind hacking PECL/mysqlnd_plugin for me?

See you at the milk bar!

Sugar!

```
class global_proxy extends mysqlnd_plugin_connection {
  public function query($conn, $query) {
    printf("%s(%s)\n", __METHOD__, $query);
    return parent::query($conn, $query);
  }
}
class specific_proxy extends mysqlnd_plugin_connection {
  public function query($conn, $query) {
    printf("%s(%s)\n", __METHOD__, $query);
    $query = "SELECT 'mysqlnd is cool'";
    return parent::query($conn, $query);
  }
}

mysqlnd_plugin_set_conn_proxy(new global_proxy());
$conn1 = new mysqli(...);
$conn2 = new PDO(...);

mysqlnd_plugin_set_conn_proxy(new specific_proxy(), $conn2);
(i_love_mysqlnd.php)
```

Sugar, sugar!

```
class proxy extends mysqlnd_plugin_connection {
  public function connect($host /*... */) {
    $conn = @parent::connect($host /*... */);
    if (!$conn) {
        my_memcache_proxy_set("failed_host", $host);
        $failover_hosts = my_memcache_proxyget("failover_hosts");
        foreach ($failover_hosts as $host) {
            $conn = @parent::connect($host /* ... */);
            if ($conn) {
                my_memcache_proxy_set("working_host", $host);
                break;
            } else {
                my_memcache_proxy_set("failed_host", $host);
            }
        }
    }
    return $conn;
  }
}
(client_failover_with_config.php)
```

Sugar, sugar Baby!

```
$pdo = new PDO(...);
$proxy = new mysqlnd_plugin_connection();
$proxy->getThreadId(mysqlnd_plugin_pdo_to_conn($pdo));

(i_do_not_love_pdo.php)
```

THE END

Contact: ulf.wendel@sun.com, Credits: Andrey Hristov