PHP class description for developers: Reading and writing with the API connection.

Name:
SalesLayer_Conn (version: 2.2)

This document explains the operations of "SalesLayer-Conn.php" class.

This class contains all the logic and connection methods to the Sales Layer customizable APIs.

Object creation:

$sl_api = **new SalesLayer_Conn** ($connector_id, $secret_key);

This object connects to the API and allow you to read and send mass data to your Sales Layer account.

Main API calls:

$sl_api->**get_info** ($last_update, $params);

Sends a connection request to the API and to prepare the information. The $last_update variable (optional) indicates the UNIX timestamp of the last API request *(this timestamp can be obtained in every request, $last_update=$sl_api->**get_response_time**('unix'), and must be stored for future API requests: $sl_api->**get_info**($last_update))*.

$params allows to send to the API any other extra parameters different from data updates.

group_category_id	By default, Saleslayer API returns one item by every category it is assigned to. If group_category_id is set to 1, every category of a product will be set inside an array. This option can be set with the function **set_group_multicategory** explained below in this document.
add_empty_categories	By default, Sales Layer API does not return categories without products assigned to them. If add_empty_categories is set to 1, the API will return all categories with visible status.

Response: **true** if everything has worked correctly.

The untreated raw data can be read via: $sl_api→data_returned.

$sl_api->**set_info** ($update_items*[, $delete_items, $compression]*);

Makes a connection request to the API which sends the data contained in **$update_items** to update them in Sales Layer. The structure contained in $update_items must be:

"**{table name}"** = [
    [
        "**{reference field}"** = "{unique item reference}"
        "{field name 1}" = *"{new value}",*
        "{field name 2}" = *"new value}",*
        ...
    ],
    … next item to update ...
],
"{next table name}" = [
    … list of items to update ...
]

It is also possible to attach the URL of a CSV file:

"**{table name}"** = "https://{CSV file URL with the data to be updated}",
"{next table name}" = "https://{CSV file URL with the data to be updated}",

Other API calls:

Rest of calls:

$sl_api->**get_response_time** ($mode);

UNIX date returned. $mode (by default "datetime") allows to indicate if you prefer date in UNIX format or "Y-m-d h:m:s" format. *Returns "false" if the request has not been done or an error occurs.*
$sl_api->**get_response_api_version** ();

Version of the API. *Returns "false" if the request has not been done or an error occurs.*

$sl_api->**get_response_action** ();

API action. *Returns "false" if the request has not been done or an error occurs.*

$sl_api->**has_response_error** ();

Checks if the API connection had an error. *Returns "**true**" if there was an error.*

$sl_api->**get_response_error** ();

Returns the error code generated by the API. *Returns "0" if there were no errors. You can find more information about error codes at the end of the document.*

$sl_api->**get_response_error_message** ();

Returns the error text. *If there was no error, returns an empty chain*.

$sl_api->**get_response_field_titles** ();

Returns the fields titles.

$sl_api→**get_response_language_field_titles** ();

Returns the fields titles in a specific language.

$sl_api->**get_response_table_information** ($table);

Returns a field list by table with the fields information (string, numeric, boolean, datetime, image or file) and if it is multi-language. The basic structure returned is the following:

```
array(
	'{tabla}' => array(
		'fields' => array(
			'{field name}' => array(
				'type' => "{string|numeric|boolean|datetime|image|file}",
				'has_multilanguage' => {0|1},
				'language_code' => '{en|es|fr|...}', ← if has_multi-language = 1
				'basename' => '{name of the field without language code}', ← if has_multi-language = 1
				'image_sizes' => array (
					'{extension}' => array ( 'width', 'height' ),
					...
				)
			),
			...
		),
		'table_joins' => array(

			'ID_{table}' => '{table}',
			...
		),
		'count_registers' => {total number of modified/deleted records},
		'count_modified' => {number of modified records},
		'count_deleted' => {number of deleted records}
```

```
        ),
        ...
)
```

$sl_api->**get_response_default_language** ();

Returns the default language code.

$sl_api->**get_response_languages_used** ();

Returns a matrix with the language codes used for multi-language fields.

$sl_api->**get_response_table_deleted_ids** ($table);

Returns a matrix with the deleted IDs of the table. ***"False"*** *if there are no deleted records.*

$sl_api->**get_response_table_modified_ids** ($table);

Returns a matrix with the modified IDs of the table. ***"False"*** *if there are no modified records.*

$sl_api->**get_response_table_modified_data** ($table);

Returns a matrix only with the modified records. ***"False"*** *if there are no modified records.*

$sl_api->**get_response_list_modified_files** ();

Returns a matrix with the images or files to download. This option is designed to be used by apps that need to know how many files are about to be downloaded. ***"False"*** *if there are no modified files.*

$sl_api->**get_response_table_data** ($table);

Returns a matrix with the information inside any table. The matrix is structured as follows:

```
array(
        'modified' => array (
                '0' => array(
                        'id' => {ID record identifier},
                        'id_parent' * => {parent ID section},
                        'id_{table name}' ** => {ID of the linked table (join)}, ← special field for nested tables
                        'data' => array(
                                {key 1} => {value 1},
                                {key 2} => {value 2},
                                ...
                        ),
                ),
                '1' => array( … ),
                …
        ),
        'deleted' => array(
                {ID 1}, {ID 2}, {ID 3}, ...
        )
```

);

* = The field 'id_parent' allows nesting hierarchically the categories tree of the catalogue, or in sales material tables.

** = The field 'ID_{table name}' only will be provided in the table 'product_formats' to link with 'products' and other nested sales material tables. The table 'product_formats' stores product variations, for instance: the different sizes, colours, fabrics... of a shirt.

$sl_api->**get_response_offline_file** ();

Returns a TAR with all the files packed. This option is intended for apps that requires an offline mode.
*Returns "**false**" if the file does not exist.*

$sl_api->**get_response_connector_Schema** ();

Returns the connetor's configuration schema. May vary in specific configurations, but usually contains:

```
array(
	'languages' => array ({codes of the languages used}),
	'default_language' => {default language},
	'offline_mode' => boolean field that indicates if a TAR file with all the files packed exists (used for apps with offline mode).
	'connector_type' => kind of connector (by default: 'default')
)
```

$sl_api->**get_response_waiting_files** ();

Return the number of images or files waiting in process:

```
array(
	'waiting' => array(
		'images' => array(
			'total' => {number of entire images in process},
			'partial' => {number of images without any size}
		),
		'files' => {number of files in process}
		)
	)
);
```

$sl_api->**set_group_multicategory** ();

Set the param group_category_id to params (0/1):

Error codes:

These error codes are shown by $sl_api->**get_response_error**() y $sl_api->**get_response_error_message**().

The following table shows the defined errors:

1	Validation error	Error
2	Invalid connector code ('code')	Error
3	Invalid unique key ('unique')	Error
4	Invalid codification key ('key')	Error
5	Date of last update incorrect	Informative, returns all data
6	The specified API version does not exist	Informative
7	Invalid output mode ('output')	Informative, returns all data in JSON
8	Invalid compression type ('compression')	Informative, does not compress.
9	Invalid private key	Error
10	Service temporary blocked	Alert
11	Service temporary not available	Alert
12	Invalid timestamp ('time')	Error
13	Expired timestamp ('time')	Error
14	Updating data. Try later	Informative
101	Empty response, incomplete or incorrectly formatted	Error
102	Network connection error	Error
103	Local database connection error	Error
104	Local database access error	Error