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

Name:
SalesLayer_Conn (version: 1.29)

Important:
Version 1.29 of SalesLayer_Conn contains updates which are incompatible with previous versions when it comes to returning lists of results.

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

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

Object creation:

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

This object connects to the API and allows 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 prepares the information. The (optional) $last_update variable indicates the UNIX timestamp of the last API request *(this timestamp is obtained with 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 you to send to the API any other extra parameters which are not data updates:

group_category_id	By default, Saleslayer API returns just one item for every category it is assigned to. If group_category_id is set to 1, all the categories of a product will be sent inside an array. This option can be set with the function **set_group_multicategory**, which is explained below.
add_empty_categories	By default, the Sales Layer API does not return categories which don't have any 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 seen via: $sl_api→data_returned.

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

Makes a connection request to the API and sends the data contained in **$update_items** so that they can be updated in Sales Layer. The structure for the data 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 calls:

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

Returns a UNIX date. $mode (by default "datetime") allows you to indicate if you prefer the date in UNIX format or in "Y-m-d h:m:s" format. *Returns "false" if the request cannot be completed or if an error occurs.*

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

Returns the version of the API. *Returns "false" if the request cannot be completed or if an error occurs.*

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

Returns the action performed by the API when generating the response. *Returns "false" if the request cannot be completed or if an error occurs.*

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

Checks if the API connection has suffered an error. *Returns "**true**" if an error has occurred.*

$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 message text. *If there was no error, it 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 per table with the information on each field (string, numeric, boolean, datetime, image or file) and whether the field is multi-language. The basic structure returned is the following:

```
[
   '{table}' =>[
      'fields' => [
         '{field name}' => [
            'type' => "{key|string|numeric|boolean|datetime|image|file|json}",
            'sanitized' => "{sanitized name for database}",
            'has_multilanguage' => {0|1},
            'title' => "Public title", ← if has_multi-language = 1
            'titles' => [ {field titles in the languages of the catalog} ], ← if has_multi-language = 0
            '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' => [
               '{extension}' => [ 'width', 'height' ],
               ...
            ]
         ],
         ...
      ],
      'table_joins' => [

         '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. Returns ***"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. Returns ***"False"*** *if there are no modified records.*

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

Returns a matrix with only with the modified records. Returns ***"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 going to be downloaded. Returns ***"False"*** *if there are no modified files.*

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

Returns a matrix with the information from the specified table. The matrix is structured as follows:

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

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

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

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

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

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

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

```
[
   'languages' => [ {codes of the languages used} ],
   'default_language' => {default language},
   'offline_mode' => boolean field that indicates if a TAR file with all the files compr 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 being or waiting to processed:

```
[
   'waiting' => [
      'images' => [
         '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 parameter group_category_id to the parameters (0/1):

Error codes:

These error codes are shown by $sl_api->**get_response_error**() and $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 temporarily blocked	Alert
11	Service temporarily 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
105	The ID code of the connector does not correspond with the provided connector	Error