PHP class description for developers: API connection and database sync

Name:
SalesLayer_Updater (version: 1.20)

This connector expands "SalesLayer-Conn.php" class.

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

Object creation:

$sl_api = **new SalesLayer_Updater** ($dbname, $dbusername, $dbpassword, $dbhost, $connector_id, $secret_key);

This object connects to the API and saves data in the local database.

Main API calls:

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

Connects to Sales Layer's API, checks if exists any data modification and updates the local database with the changes. This object also creates the tables that will store data.

$sl_api->**extract** ($table, $fields, $language, $conditions);

Runs data readings on the tables of the local database. This object is responsible for responding to the data in any required language. You can also build complex selection clauses to search.

Examples:

```
$files = array('section_name', 'section_description');

$conditions=array(
	array(
		'field' =>'section_description',
		'search'=>'red'
	),
	array(
		'field'    =>'parent_section_reference',
		'condition'=>'!=',
		'value'    =>'RF45'
	)
);

$datos = $SLupdate->extract('catalogue', $files, 'en', $conditions);
```

Other API calls:

$sl_api->**database_connect** ($database, $username, $password, $hostname)

Connects with the local database.

$sl_api->**get_database_table_schema** ($table, $extended);

Returns the data schema of the stored information.

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

Returns any tables linked to any given table.

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

This call checks if any table returned by the API exists in the local database.

$sl_api->**create_database_table** ($table, $schema);

Creates a table in the local database based on the schema returned by the API.

$sl_api->**update_database_table** ($table, $schema);

Updates the table with a new modified schema.

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

Returns a list with all the IDs of the records stored in a table of the local database.

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

Returns the default language.

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

Returns a list of available languages.

$sl_api->**update_database_table_data** ($table, $data);

Updates a local database table with the new records.

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

Deletes all the data and tables imported.

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

Shows in a matrix all the calls to the database.

$sl_api->**delete_connector** ($code, $clean_items=false);

It removes a connector and all associated items in the database. If you only want to delete the reference connector but keep the items the second parameter ($clean_items) must be marked as true.

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

Print debugging information if this is active.

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

Returns the titles of the field.

$sl_api→**get_language_field_titles** ();

Returns the titles of the fields in a specific language.

Rest of calls (inherited from **saleslayer_conn**) class:

$sl_api->**get_response_time** ($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 occurs an error.*

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

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

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

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

$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*.

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

Sends a connection request to the API and prepares results. The variable $last_update (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 (to be defined in future...).

Response: **true** if everything went well.

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

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

Returns the titles of the field.

$sl_api→**get_response_language_field_titles** ();

Returns the titles of the fields in a specific language.

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

Returns a list of fields by table with the information of the field (string, numeric, boolean, datetime, image or file) and if the field 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_multilanguage = 1
                'basename' => '{name of the field without language code}', ← if has_multilanguage = 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 example: the different sizes, colours, fabrics... of a shirt.

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

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

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

Returns the configuration schema of the connector. May change on very specific configurations, but basically 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}
                )
        )
);
```

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 mode.
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