CAL Tables.

The CAL API python scripts or programs are separate from the main programs designed to deal with the reporting API. All scripts that work on the CAL API are prefixed by “cal_”. These use some functions from the main scripts.

The scripts rely on a series of database tables for their operation. The most important of these is the prog_ix_api_to_db_table_mapping which maps APIs to tables which store the data recovered from those APIs. The authentication for the CAL API works differently to that for the reports API and needs an API key instead of a email address and associated password. The authentication table is also important to get the API keys from. Each of the APIs has one table. In addition, the campaign mapping table has a sub table (NB this is not used at the time of writing as an API key that works with the campaign mapping table has not been supplied).

The purpose of each of the files is listed below:

File	Content
cal-create-db-tables.py	Utility program to set up the CAL tables in the database, and populate the API table. Comments To use this you must set the environmental variables PROGSALESMYSQLUSERNAME PROGSALESMYSQLPASSWORD To be a username and password that has permission to create tables as by default iximport the default user does not. This use should also not have a “caching_sha2_password“ password as the mysql connector does not support this password type. You will also need to populate the authentication table with APIKEYs with the add-api-key.py program.
cal_log_retrieval_API.py	Functions to call the REST API for the log retrieval APIs
cal_mappings_service_API.py	Functions to call the REST API for the mappings service APIs
cal_authenticationAPI.py	Functions to perform authentication to Index Exchange for CAL APIs.
cal_utility.py	Various utility functions to support other files.
cal_settings.py	Contains settings for the CAL downloads
cal_mysql.py	MySQL functions specific to the CAL files
calcoreAPI.py	Functions common to cal_*_API files

cal_exchange_rates_service_API.py	Functions to call the REST API for the exchange rates service APIs Not currently used, but work.
cal-download.py	The “main” CAL API program which uses all other file here to do its work of downloading data via the CAL APIs and uploading the retrieved data to the MySQL database.
create-api-keys.py	Utility program to insert keys into the CAL API auth database (the prog_ix_apikeys table).
delete-api-key.py	Utility program to delete a publisher_id and associated key from the the prog_ix_apikeys table.

Operation of the cal-download.py program

In operation as each publisher has its own API key and the output of the Log APIs, the campaign, and deal APIs depend on the publisher (and hence the publiher_id and hence API KEY used) the data from the API is augmented with the publisher_id used to obtain the data. The information returned from the other CAL APIs (i.e. the buyer, DSP, site, and partner) is the same for each publisher and so only the very first publisher in the authentication tables API key is used to download this information (and it is not augmented with the publisher_id)

The operation of the download program is simple. It takes in turn each key from the API Key table (the prog_ix_apikeys table) obtains an authentication token and then downloads the mappings in the table below. These are the mappings that provide different data which depends on the publisher (and hence publisher_id). The data obtained is placed in the data base table as specified by the entry for the API in the prog_ix_api_to_db_table_mapping.

API	Purpose	Comment
campaign_mapping	Mapping of campaigns	
deal_mapping	Mapping of deals	
site_mapping	Mapping of sites	

Table 1

For the very first publisherID in the the prog_ix_apikeys table, that publishers API key is used to download data from the following APIs. The data supplied by these APIs is invariant to publisher_id and hence only the very first publisher_id is used to obtain this data. The data obtained is placed in the data base table as specified by the entry for the API in the prog_ix_api_to_db_table_mapping.

API	Purpose	Comment
Buyer_mapping	Mapping table of buyers	
dsp_mapping	Mapping table of DSPs	
partner_mapping	Mapping table of partners	

Table 2

Then if the program is called with one of the -i, -b, and -l flags the appropriate log tables are downloaded and transformed and uploaded to the table in the database. The flags determine if the log tables are downloaded and their effect is given in the table below:

Flag	effect
-M	Do not download the mapping tables
-i	Download the impressions logs
-b	Download the bid logs
-l	Download both the impression and bid logs
-D	Show, voluminous, debugging information

The code contains functionality such that prior to uploading the data to the database it can be transformed in some way by the internal transform mapping. This is specified in the downloads_mapping_service and download_log_service Python dictionary structures.

This is used to add a publisher_id to the data from the mappings in table 2 above and to the data from the log tables. It is also used to flatten the section_id element of the deal mapping table and flatten the creative_id in the campaign_mapping (it was intended to be used to create the sub table for the campaign table the creatives table). The program internally maintains a record of all file downloads from the mapping and log APIs in the prog_ix_cal_download_log table.

Utility programs

The following utility programs are provided as part of the CAL API suite:

Program name	purpose
add-api-key.py	Adds or replaces a publisher_id and apikey to the apikey table If run with no parameters will display the current apikey table Parameters are pairs of punlisher_id and apikey. The API KEYs is in hex Example

	python3 add-api-key.py 1234 “shjcxwhXAFHE12cDs” 234 “sidjcisdjVHRH” Will add or update the key “shjcxwhXAFHE12cDs” for the bublisher_id 1234, and will add or update the key “sidjcisdjVHRH” for the publisher_d 234.
delete-api-key	Takes a list of publisher_ids and deletes any entry with those publisher_ids from the api key table. If run with no parameters will display the current API KEY table. Parameters are the publisher_ids of the publisher to remove. Example python delete-api-key 123455 24832 Will delete the key associated with publisher 123455 and 24832 from the API KEY database.

Database Tables

This section details the database tables used by the programs.

Program support tables

These tables are needed for the cal-download.py program to operate.

Authentication table

This table stores the api key for each publisher. Like all other tables detailed here it is held on the MySQL server.

```
create table prog_ix_apikeys (
      publisher_id integer NOT NULL,
      api_key varchar(255) NOT NULL,
      PRIMARY KEY(publisher_id)
);
```

To grant access to the table

```
grant select,update,insert on prog_stats.prog_ix_apikeys to
iximport@localhost ;
```

The API to table mapping table

This table stores a map of API functions calls to table. This is needed as the system needs to know which database table to store the output of each of the APIs as they provide very different data and structures.

```
create table prog_ix_api_to_db_table_mapping (
      api varchar(60) NOT NULL,
      data_table varchar(60) NOT NULL,
      download_log_table varchar(60),
      frequency varchar(1) NOT NULL,
      PRIMARY KEY(api)
);
```

To grant access to the table

```
grant select,update,insert on
prog_stats.prog_ix_api_to_db_table_mapping to iximport@localhost;
```

The fields `download_log_table` and `frequency` are historic and no longer used.

CAL log API tables

These are database tables that are used to store information about Index Exchange impression and bid log information.

Impression event log table

This stores information from Index Exchange about impressions.

```
CREATE TABLE prog_ix_impression_log (
     auction_id varchar(36) NOT NULL default '',
     publisher_id int NOT NULL ,
     timestamp timestamp NULL DEFAULT NULL,
     country varchar(2) DEFAULT '',
     state varchar(16) DEFAULT '',
     dma int NOT NULL,
     zip_postal varchar(32) DEFAULT '',
     device_type varchar(255) DEFAULT '',
     operating_system varchar(255) DEFAULT '',
     browser varchar(16) DEFAULT '',
     partner_id int NOT NULL,
     site_id int NOT NULL ,
     creative_type varchar(16) DEFAULT '',
     domain varchar(255) DEFAULT '',
     app_bundle varchar(255) DEFAULT '',
     inventory_channel varchar(255) DEFAULT '',
     supply_source varchar(255) DEFAULT '',
     slot_id int NOT NULL,
     size varchar(255) DEFAULT '',
     event_type varchar(255) DEFAULT '',
     event_opportunity varchar(255) DEFAULT '',
     gross_revenue decimal(10,5) NOT NULL DEFAULT '0.00000',
     net_revenue decimal(10,5) NOT NULL DEFAULT '0.00000',
     dsp_id int NOT NULL,
     trading_desk_id int NOT NULL,
     campaign_id int NOT NULL,
     campaign_name varchar(255) DEFAULT '',
     brand_id int NOT NULL,
     brand_name varchar(255) DEFAULT '',
     creative_id int NOT NULL,
     creative_name varchar(255) DEFAULT '',
     deal_id varchar(255) NOT NULL,
     internal_deal_id int NOT NULL,
     cookie_match_status varchar(255) DEFAULT '',
     a_domain varchar(255) DEFAULT '',
     pub_rev_share decimal(10,5) NOT NULL DEFAULT '0.00000',
     billing_term_id int NOT NULL,
     p_ids varchar(255) DEFAULT '',
     discount_amount decimal(10,5) NOT NULL DEFAULT '0.00000',
     video_placement_type varchar(255) DEFAULT '',
     other varchar(255),
     PRIMARY KEY (auction_id)
     );"
```

To grant access to the table

```
grant select,update,insert on
prog_stats.prog_ix_impressions_event_fileid to iximport@localhost ;
```

The bid event table

This table contains information downloaded from the bid event API. It contains information about bids.
Note this is the same structure as the impressions event table.

```
CREATE TABLE prog_ix_bid_log (
     auction_id varchar(36) NOT NULL default '',
     publisher_id int NOT NULL ,
     timestamp timestamp NULL DEFAULT NULL,
     country varchar(2) DEFAULT '',
     state varchar(16) DEFAULT '',
     dma int NOT NULL,
     zip_postal varchar(32) DEFAULT '',
     device_type varchar(255) DEFAULT '',
     operating_system varchar(255) DEFAULT '',
     browser varchar(16) DEFAULT '',
     partner_id int NOT NULL,
     site_id int NOT NULL ,
     creative_type varchar(16) DEFAULT '',
     domain varchar(255) DEFAULT '',
     app_bundle varchar(255) DEFAULT '',
     inventory_channel varchar(255) DEFAULT '',
     supply_source varchar(255) DEFAULT '',
     slot_id varchar(255) NOT NULL,
     size varchar(255) DEFAULT '',
     event_type varchar(255) DEFAULT '',
     event_opportunity varchar(255) DEFAULT '',
     gross_revenue decimal(10,5) NOT NULL DEFAULT '0.00000',
     net_revenue decimal(10,5) NOT NULL DEFAULT '0.00000',
     dsp_id int NOT NULL,
     trading_desk_id int NOT NULL,
     campaign_id int NOT NULL,
     campaign_name varchar(255) DEFAULT '',
     brand_id int NOT NULL,
     brand_name varchar(255) DEFAULT '',
     creative_id int NOT NULL,
     creative_name varchar(255) DEFAULT '',
     deal_id varchar(255) NOT NULL,
     internal_deal_id int NOT NULL,
     cookie_match_status varchar(255) DEFAULT '',
     a_domain varchar(255) DEFAULT '',
     pub_rev_share decimal(10,5) NOT NULL DEFAULT '0.00000',
     billing_term_id int NOT NULL DEFAULT '-9999',
     p_ids varchar(255) DEFAULT '',
     discount_amount decimal(10,5) NOT NULL DEFAULT '0.00000',
     video_placement_type varchar(255) DEFAULT '',
     other varchar(255),
     PRIMARY KEY (auction_id)
     );"
```

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_bid_event_fileid to
iximport@localhost ;
```

The event log download table

In addition to the impressions event log table and the bid event log table a record of each file downloaded from Index Exchange via the impression event API is kept in the impression event download log table.

```
create table prog_ix_cal_download_log (
      fileid integer NOT NULL,
      hour datetime NOT NULL,
      downloadURL varchar(255) NOT NULL,
      publisher_id integer NOT NULL,
      origin varchar(255) NOT NULL,
      PRIMARY KEY(fileid, publisher_id)
```

```
     );
```

Origin is one of “impressions_log”, “bid_log”.

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_cal_download_log to
iximport@localhost;
```

Tables to store information about the mappings service in

Tables to manage the storage of mapping service data obtained by the mapping service APIs.

Buyer mapping table

```
create table prog_ix_buyer_mapping (
     dsp_id integer NOT NULL,
     trading_desk_id integer NOT NULL,
     buyer_name varchar(255) NOT NULL,
     PRIMARY KEY (dsp_id,trading_desk_id)
     );
```

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_buyer_mapping
to iximport@localhost ;
```

The campaign mapping table

```
create table prog_ix_campaign_mapping (
     brand_id integer NOT NULL,
     brand_name varchar(255) NOT NULL,
     campaign_id integer NOT NULL,
     campaign_name varchar(255) NOT NULL,
     creative_id int NOT NULL,
     publisher_id integer NOT NULL,
     PRIMARY KEY (brand_id,campaign_id),
     FOREIGN KEY (creative_id) REFERENCES
prog_ix_creatives(creative_id)
     );
```

To grant access to the table

```
grant select,update,insert on
prog_stats.prog_ix_campaign_mapping to iximport@localhost ;
```

The creatives table

This table is needed as the campaign API returns a list as a data element and this cannot be conveniently stored in a flat table. Therefore the table is separated out.

```
create table prog_ix_creatives (
     creative_id integer NOT NULL,
     creative_name varchar(255) NOT NULL,
     PRIMARY KEY (creative_id)
     );
```

To grant access to the table

```
grant select,update,insert on prog_stats.prog_ix_creatives to
iximport@localhost ;
```

This table is not currently used.

The deal mapping table

```
create table prog_ix_deal_mapping (
     deal_id varchar(255) NOT NULL,
     internal_deal_id integer NOT NULL,
     deal_name varchar(255) NOT NULL,
     dsp_id integer NOT NULL,
     deal_type integer NOT NULL,
     auction_type varchar(255) NOT NULL,
     rate float NOT NULL,
     section_id varchar(255) NOT NULL,
     status char(1) NOT NULL,
     start_date date NOT NULL,
     end_date date NOT NULL,
     priority integer NOT NULL,
     publisher_id integer NOT NULL,
     PRIMARY KEY (internal_deal_id)
     );
```

To grant access to the table

```
grant select,update,insert on prog_stats.prog_ix_deal_mapping to
iximport@localhost ;
```

Note that the section_id is flattened from a list to a string as it is not used at Prog Sales.

The DSP mapping table

```
create table prog_ix_dsp_mapping (
     dsp_id integer NOT NULL,
     dsp_name varchar(255) NOT NULL,
     PRIMARY KEY(dsp_id)
);
```

To grant access to the table

```
grant select,update,insert on prog_stats.prog_ix_dsp_mapping to
iximport@localhost ;
```

The site mapping table

```
create table prog_ix_site_mapping (
     site_id integer NOT NULL,
     site_name varchar(255) NOT NULL,
     PRIMARY KEY(site_id)
);
```

This is run for all publishers.

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_site_mapping to
iximport@localhost ;
```

The partner mapping table

```
create table prog_ix_partner_mapping (
     partner_id integer NOT NULL,
     partner_name varchar(255) NOT NULL,
     PRIMARY KEY(partner_id)
);
```

This is run for all publishers.

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_partner_mapping
to iximport@localhost ;
```

Tables to store information about the exchange rates service in

This table is designed to be used with the exchange rate mapping API. However, this information is not used at Prog Sales and so is not used and not implemented.

The exchange rates mapping table

```
create table prog_ix_exchange_rates (
     base varchar(255) NOT NULL,
     currency varchar(255) NOT NULL,
     Date NOT NULL,
     Rate float NOT NULL,
     PRIMARY KEY(partner_id)
);
```

To grant access to the table

```
grant select,update,insert on  prog_stats.prog_ix_exchange_rates
to iximport@localhost ;
```