Digital Travel Marketing Group
Suite 104 to 105
The Cotton Exchange
Liverpool
United Kingdom
L3 9LQ

CruiseAppy Technical Specification

TravelTek API

Last Update: Mar 13, 2025
By: Peter Aland

Introduction

This document covers the CruiseAppy plugin for the TravelTek XML API.

Requirements

Server

The CruiseAppy plugin has been fully tested with Ubuntu 22.04 with Apache and Nginx. Other environments may work but have not been fully tested.

Application, Software and Plugins

The CruiseAppy plugin requires the following pre configured elements to work correctly:

Element	Version
PHP	8.4
MySQL	8.4
Xmlstarlet	1.6.1
JQ	1.7
WordPress	6.7
-Advanced Custom Fields Pro	6.3.12

Scripts (bourne again shell)

The CruiseAppy plugin uses bash scripts for automation tasks relating to importing and updating data from the TravelTek XML API.

config.sh

This script sets up the necessary directories and retrieves configuration options from a WordPress database.

Directories:

- Creates the following directories within the script path:
  - Csv
  - Json
  - Log
  - Traveltek
  - xml
- Creates the following directories within the WordPress path:
  - Log
  - volume

.htaccess and .htpasswd:

- Creates an .htaccess file in the xml directory to restrict access.
- Creates an .htpasswd file in the xml directory with a predefined user and password.

Database Connection:

- Retrieves the database name and table prefix from the WordPress configuration.
- Determines the database connection method based on the database type.

WordPress Options:

- Retrieves various options from the WordPress database and assigns them to variables.
- The options include settings for the application, at sea mode, cabin selection, payment gateway, search, smart package, currency, endpoints, FTP credentials, language, market ID, packages, site name, status, username, website URL, WordPress path, WPML, payment passthrough, and booking endpoint.
- Some options have default values if not set in the database.

control-import.sh

This script performs the following tasks:

1. Checks if the script is already running to prevent multiple instances.
2. Deletes old WordPress posts of type 'cruiseappy-imports' older than 7 days.
3. Generates a unique import ID.

4. Ensures the log file exists, creating it if necessary.
5. Fetches the list of files from an FTP endpoint.
6. Identifies and downloads the latest file from the FTP endpoint.
7. Unzips the downloaded file.
8. Removes log entries older than 7 days from the database.
9. Processes 'simplesearch' XML files to count <cruise> nodes and logs the data into the database.
10. Creates a new WordPress post with the import report using WP-CLI.
11. Adds meta data to the WordPress post for each processed file.
12. Checks for files dated in the next 2 years with 0 cruises and updates the import status accordingly.
13. Performs various cleanup and setup tasks, including creating directories and rotating logs.
14. Fetches the latest file from the TravelTek FTP server and processes it.
15. Backs up any booking XML files.
16. Runs several control scripts for processing cruise lines, ships, ports, and other data.
17. Protects prefetch prices and starts the prefetch prices script in the background.
18. Updates the search index and logs the status.
19. Sends an email notification upon completion of the import process.
20. Updates the import status to 'Completed' in the WordPress post.

control-cruiselines.sh

This script controls the import and update of cruise lines data from the TravelTek XML API into a WordPress site.

It performs the following steps:

1. Sets the script path and sources the functions.sh file.
2. Defines the log file path.
3. Checks if the script is already running to prevent multiple instances.
4. Removes the existing cruiselines.xml file.
5. Calls the control-tables.sh script to prepare the database table.
6. Constructs the XML request to get cruise lines data from the external API.
7. Sends the request using curl and saves the response to cruiselines.xml.
8. Checks for errors in the XML response and logs them if any.

9. Renames XML attributes for consistency.
10. Loads the XML data into the MySQL database table.
11. Updates the database table with post IDs from WordPress.
12. Removes locked cruise lines from the updates.
13. Selects cruise lines without post IDs and creates new WordPress posts for them.
14. Imports the logo URL as the featured image for the new posts.
15. If WPML is active, creates translations for the new posts.
16. Calls the update-cruiseline.sh script to update the new posts.

control-ships.sh

This script controls the import and update of ships data from an external API into a WordPress site.

It performs the following steps:

1. Sets the script path and sources the functions.sh file.
2. Defines the log file path.
3. Removes the existing ships.xml file.
4. Calls the control-tables.sh script to prepare the database table.
5. Constructs the XML request to get ships data from the external API.
6. Sends the request using curl and saves the response to ships.xml.
7. Renames XML attributes for consistency.
8. Loads the XML data into the MySQL database table.
9. Updates the database table with post IDs from WordPress.
10. Removes locked ships from the updates.
11. Selects ships without post IDs and creates new WordPress posts for them.
12. If WPML is active, creates translations for the new posts.
13. Calls the update-ship.sh script to update the new posts.

control-flat-files.sh

This script processes and imports XML data files into a MySQL database for a WordPress site.

It performs the following tasks:

1. Sets the script path and sources a functions file.
2. Applies a hotfix for the Admin Columns Pro plugin.
3. Checks if the script is already running to prevent multiple instances.

4. If WPML is installed, it retrieves file languages and default languages from the database.
5. Ensures the file_language variable is set.
6. Loops through the JSON array of file languages and processes each language:
    a. Deletes existing data for the current language from several database tables.
    b. Sets the source folder based on the file language.
    c. Processes XML files in the source folder and imports data into the database.
    d. Imports itineraries for the current language.
7. Updates the database with ship, cruise line, and post ID information.
8. Deletes outdated or invalid cruise data.
9. Trashes WordPress posts without a corresponding **codetocruiseid** in the TravelTek cruises table.

control-cruises.sh

This script performs various database operations related to cruise data.

It includes the following steps:

1. Determine the script's directory and source the functions.sh file.
2. Define the log file path.
3. Check if the script is already running to prevent multiple instances.
4. Execute a series of MySQL commands to update and manage cruise data:
    a. Update the post_type in the ff_traveltek_cabingrades table.
    b. Update the post_id in the ff_traveltek_cruises table based on the ca_wpml flag.
    c. Update prices in the ff_traveltek_cruises table.
    d. Update various meta values in the WordPress postmeta table.
    e. Insert new records into the ff_traveltek_cabingrades table for different cabin types and prices.

Note: Some shellcheck warnings are disabled for specific lines.

control-new-cruises.sh

This script is used to update and create new cruise posts in a WordPress site.

It performs the following tasks:

1. Checks if the script is already running to prevent multiple instances.
2. Logs the start time of the script.
3. Executes MySQL queries to update and select data from the `ff_traveltek_cruises` table.
4. For each cruise that needs a new post:
    a. Checks if the cruise ID already exists in the `postmeta` table.
    b. If not, creates a new WordPress post with the cruise details.
    c. Inserts post meta data into the database.
    d. Updates the `post_id` in the `ff_traveltek_cruises` table.
    e. Deletes existing cabin grades for the new post.
    f. Inserts new cabin grades for the new post.
    g. If WPML is enabled, creates translations for the new post.
    h. Runs the itinerary update script for the new post.
5. Logs the end time of the script.

control-ports.sh

This script controls the ports for the CruiseAppy application. It performs the following tasks:

1. Sets the script path and sources the functions.sh script.
2. Logs the start of the control ports process.
3. Executes the control-tables.sh script with the argument ff_traveltek_ports.
4. Logs the current date and ca_tt_language variable.
5. If ca_wpml is set to 1, it performs the following:
    a. Queries the database for cruise IDs and their corresponding languages.
    b. For each cruise ID and language, it determines the folder and file path.
    c. If the file exists, it loads the XML data into the ff_traveltek_ports table.
    d. If the file does not exist, it deletes the corresponding entry from the ff_traveltek_ports table.
    e. Logs the process details.
    f. If ca_wpml is not set to 1, it performs the following:
    g. Queries the database for cruise IDs.
    h. For each cruise ID, it determines the folder and file path.
    i. Loads the XML data into the ff_traveltek_ports table.

    j. Updates the ff_traveltek_ports table with port_post_id and cruiseline_post_id based on the postmeta and posts tables.
    k. Queries the ff_traveltek_ports table for distinct port_lang_line_id, language, portid, and lineid where port_post_id is NULL.
6. For each result, it performs the following:
    a. Logs the port details.
    b. Creates a new WordPress post for the port.
    c. Inserts metadata for the new post into the postmeta table.
    d. If ca_wpml is set to 1, it handles the translation of the post.
7. Logs the end of the control ports process.

control-incremental.sh

This script performs an incremental update of itineraries for a WordPress site.

It checks if the script is already running and exits if it is.

It sources a functions script and then performs different operations based on the value of the `ca_wpml` variable

If `ca_wpml` is set to "1", it performs a WPML incremental itinerary update.

Otherwise, it performs a standard incremental itinerary update.

The script does the following:

1. Checks if the script is already running and exits if it is.
2. Logs the start time.
3. If `ca_wpml` is set to "1":
    a. Retrieves languages from the database and processes each language.
    b. Truncates the `ca_itinerary_lookup` table.
    c. Inserts itinerary data into the `ca_itinerary_lookup` table.
    d. Updates the `ca_itinerary_lookup` table with post IDs and live itineraries.
    e. Sets the itinerary status to 1 where the itinerary differs from the live itinerary.
    f. Retrieves the count of posts with itinerary status 1.
    g. Limits the number of posts to update based on the count.

    h. Updates itineraries for the posts in batches of 3, running the updates in the background.
4. If `ca_wpml` is not set to "1":
    a. Logs the start time.
    b. Truncates the `ca_itinerary_lookup` table.
    c. Inserts itinerary data into the `ca_itinerary_lookup` table.
    d. Updates the `ca_itinerary_lookup` table with post IDs and live itineraries.
    e. Sets the itinerary status to 1 where the itinerary differs from the live itinerary.
    f. Updates itineraries for the posts in batches of 10, running the updates in the background.

control-search.sh

This script performs various operations related to cruise data management in a WordPress environment.

It includes tasks such as checking if the script is already running, logging, managing MySQL tables, updating search tables, handling WPML languages, updating cruise line and ship information, managing cruise prices, and more.

The script performs the following main tasks:

1. Check if the script is already running and exit if it is.
2. Log the start of the script execution.
3. Execute control-tables.sh scripts for various lookup tables.
4. Delete prices older than 48 hours from the database.
5. Setup search tables by creating and populating them with cruise data.
6. Handle WPML languages if enabled.
7. Update cruise line and ship information in the search table.
8. Update cruise subtitles, extras, codes, and IDs.
9. Update itinerary information including embark and disembark ports.
10. Update destination information and create a destination lookup table.
11. Update local departure information.
12. Update ship information including star ratings and links.
13. Update cruise duration and duration groups.
14. Update travel types and handle special travel types.
15. Prefetch data for cruises.
16. Update cruise prices from various sources including flat files, packages, and specials.

17. Calculate and update minimum and maximum prices.
18. Handle "call for price" cruises if enabled.
19. Update best cabin types, best value, and most popular cruises.
20. Update ship and cruise line image URLs.
21. Hide cruises from search based on meta key.
22. Update cabin types for ships.
23. Update display titles for cruises based on various options.
24. Handle cruise packages if enabled.
25. Drop and rebuild search and destination tables.
26. Flush WordPress cache.
27. Log the end of the script execution.

control-prefetch-cache.sh

This script manages the prefetch cache for the CruiseAppy application. It ensures that the cache is updated with the latest cruise data by performing the following steps:

1. Checks if the script is already running to prevent multiple instances.
2. Logs the start of the prefetch process.
3. Truncates the `ca_prefetch` table in the database.
4. Reads post IDs from the database and processes each one to fetch cruise IDs.
5. For each cruise ID, checks the last updated time and updates the `ca_prefetch` table if necessary.
6. Logs the total number of prefetches.
7. Processes the random sailings with a pause between each to avoid overloading the system.
8. Runs the `live-pricing-post-id.sh` script concurrently, limiting to 3 instances at a time.
9. Waits for all background jobs to finish.
10. Logs the completion of the prefetch process.

control-tables.sh

This script is used to create and manage various tables in the MySQL database for the CruiseAppy application.

It takes a table name as an argument and creates the specified table or all tables if "all" is passed as the argument.

Usage:

- ./control-tables.sh <tablename>

Arguments:

- tablename: The name of the table to create. Use "all" to create all tables.

The script performs the following actions:

- Sets the script path and sources the functions.sh file.
    - Logs the start of the table creation process.
    - Creates the following tables in the specified database:
    - Ff_traveltek_basket_cruise
    - Ca_prefetch
    - Ff_traveltek_basket_items
    - Cabin_select_itinerary
    - Search_destinations
    - Search_destinations_temp
    - Search_cruiseline_lookup
    - Search_itinerary_count
    - Search_itinerary_disembark
    - Search_itinerary
    - Search_ship_lookup
    - Search_new
    - Ff_traveltek_ratecodes
    - Searches
    - Ff_traveltek_cabingrades_temp
    - Ff_traveltek_cabingrades
    - Ff_traveltek_itinerary
    - Ff_traveltek_tms
    - Ff_traveltek_destinations
    - Ff_traveltek_packages
    - Ff_traveltek_package_itinerary
    - Ff_traveltek_package_flights
    - Ff_traveltek_package_base_price
    - Ff_traveltek_cruiselines
    - Ff_traveltek_ship_cabins
    - Ff_traveltek_ship_content
    - Ff_traveltek_ship_decks
    - Ff_traveltek_ship_details
    - Ff_traveltek_ship_images
    - Ff_traveltek_ships
    - Ff_traveltek_prices
    - Ff_traveltek_cruises

    - Ff_traveltek_cruiseline_lookup
    - Ff_traveltek_ports
    - Ff_traveltek_cruiseline_content
    - Ff_image_sizes
    - Ff_traveltek_package_cabins
    - Ff_traveltek_itinerary_image
    - Ff_traveltek_package_hotel_rooms
    - Ff_traveltek_package_transfers
    - Ff_traveltek_basket_searches_cruise
    - Ff_traveltek_basket_searches
    - Ff_traveltek_airports
    - ff_traveltek_airports_iata
- Logs the creation of each table.

Variables

Variable	Description
script_path	The path to the directory containing the script.
source_folders	The folders containing the XML files to be processed.
file_language	The language of the files being processed.
default_lang	Indicates if the language is the default language.
ca_db_name	The name of the MySQL database.
ca_db_table_prefix	The prefix for the WordPress database tables.
ca_wpml	Indicates if the WPML plugin is installed.
ca_cabin_select	Indicates if the cabin select feature is enabled.

functions.sh

This script contains various functions used in the CruiseAppy plugin for WordPress.

It includes functions for obfuscating credentials, debugging, error checking, creating and deleting temporary tables, updating search prices, logging, and more.

Variables:

- script_path: The path to the current script.
- ca_website_url: The URL of the website.
- ca_tt_password, ca_tt_username, ca_tt_sitename: Credentials for accessing the TravelTek API.
- wordpress_path: The path to the WordPress installation.
- ca_db_name: The name of the database.
- ca_db_table_prefix: The prefix for the database tables.
- ca_tt_language: The language code for WPML.
- ca_prices_fallback: A flag indicating whether to use fallback prices.
- ca_tt_sid: The session ID for holiday searches.
- ca_tt_flight_sid, ca_tt_hotel_sid, ca_tt_flight_and_hotel_sid: Session IDs for different search types.
- ca_cabin_select_holding_image: The ID of the holding image for cabins.
- ca_cabin_select_cruiseline_*: Various credentials and settings for cruise lines.
- ca_tt_mmb_endpoint: The endpoint for the TravelTek MMB API.

Functions:

- obfuscate_credentials(file): Obfuscates the credentials in the specified file.
- debug(search_id, post_id): Displays debug information and stores results in the database.
- error_check_response(response, search_id, error_message): Checks for errors in the API response and logs them.
- cabingrades_temp_table_create(post_id): Creates a temporary table for cabin grades.
- cabingrades_temp_table_delete(post_id, farecode): Deletes the temporary table for cabin grades and inserts data into the main table.
- flat_file_pricing(search_id): Inserts flat file pricing data into the database.
- update_best_price(post_id, sessionkey): Updates the best price for the specified post.
- update_search_price(): Updates the search prices in the database.
- get_ship_post_id(ship_code): Retrieves the post ID and name of the ship based on the ship code.

- log_and_price_rotate(): Rotates the log files and creates a backup of the database.
- holiday_search_sid(search_type): Sets the session ID for holiday searches based on the search type.
- get_ship_detail_image_name(image_item, image_name): Retrieves the ship detail image and saves it locally.
- cs_get_token(line_id): Retrieves the token for the specified cruise line.
- vmb_post_id(bookingid, bookingreference): Retrieves or creates the post ID for the specified booking.
- wp_add_portfolio_details(bookingid, post_id): Adds portfolio details to the specified post.
- wp_add_portfolio_documents(bookingid): Adds portfolio documents to the specified booking.
- wp_add_portfolio_cruise(bookingid, post_id): Adds cruise details to the specified booking.
- wp_add_portfolio_passengers(bookingid, post_id): Adds passenger details to the specified booking.
- wp_add_package_details(bookingid, file, post_id): Adds package details to the specified booking.
- mysql_portfollio_details(bookingid, file): Inserts portfolio details into the database from the specified XML file.