

How to Create a Time Sync Plus (TS+) Database

This guide outlines the steps to generate a local SQLite database for the Time Sync Plus (TS+) application. The database stores polygon data, user-attributed events, and UI display configurations for attribution inputs.

You will:

- Set up a Python environment
- Create a configuration file
- Run a Python script to generate the database

Prerequisites

Before getting started, ensure you have:

- Python 3.8 or higher
- Conda or Miniconda installed
- A working TS+ project folder structure
- Basic command-line knowledge

Step 1: Create and Set Up Your Conda Environment

Create and activate a Conda environment:

```
conda create -n tspo_env python=3.10 -y
conda activate tspo_env
```

Install the required packages (mostly standard libraries):

```
conda install -y -c conda-forge sqlite python=3.10
```

The following built-in libraries are used in the database creation script:

```
import json
import re
import sqlite3
from sqlite3 import Error
from sys import platform
from collections import OrderedDict
import os
import glob
import sys
import itertools
import random
```

No additional third-party libraries are required.

Step 2: Understand the Database Tables

The database includes three core tables:

Table	Description
<code>polygonTable</code>	Stores the polygon geometry and metadata. These are the plots being attributed.
<code>displayTable</code>	Defines the UI layout, labels, and interaction type (e.g., dropdowns, sliders).
<code>eventTable</code>	Captures the user's attribution input and associated metadata.

Step 3: Edit the Configuration File

Your `config.json` defines the structure of the database. Here's an example:

```
{
  "polygonTable": ["plotid", "tracker", "reEval", "user1", "user2",
    "geo", "json"],
```

```

"displayTable": {
  "Probability_Forest_Loss": ["Prop", "loss", "Stable", "Growth"],
  "Image_Condition": ["Prop", "Hinder", "NAN", "Helped"],
  "Land_Class": ["Water", "Barren", "Forest", "Agland", "Urban",
"Shrubland", "Grassland"],
  "Confidence": ["Dial"],
  "Comment": ["comment"],
  "Valid_Name": ["comment"]
},
"eventTable": [
  "plotId", "Valid_Date", "LT_YOD",
  "Probability_Forest_Loss", "Image_Condition", "Spatial_Context",
  "Valid_Name", "Confidence", "Comment",
  "User_IP", "user_elapsed_time"
]
}

```

Important: Do not change the structure of `polygonTable` or the first three fields in `eventTable`, as they are required by the application backend.

Avoid special characters in field names (e.g., spaces, punctuation). Use underscores instead.

Button Types in `displayTable`

Each entry in the `displayTable` defines a UI input used for plot attribution. The TS+ interface supports the following types:

Input Type	Description
<code>comment</code>	Opens a text box for freeform user responses.
<code>Prop</code>	Displays a proportional box for rating multiple categories.
<code>Dial</code>	Shows a knob-style input for rating confidence or similar scalar values.
List of labels (e.g., <code>"Water"</code> , <code>"Urban"</code> , <code>"Forest"</code>)	Generates a dropdown menu for selecting one of the listed options.

Default Dropdown Behavior

When no input type (e.g., `Prop`, `Dial`, or `comment`) is explicitly specified and only label options are listed (e.g., `["Forest", "Urban", "Water"]`), the system defaults to a dropdown menu. This dropdown allows the user to select one label from the list. This is the case for `Land_Class` in the example above.

Step 4: Run the Database Creation Script

Run the Python script to generate the database:

```
python 01_geojsonToDB.py <path/to/project_directory> <project_name>
```

Example:

```
python 01_geojsonToDB.py /vol/v1/TSP0/data/tspo_expo/ tspo_expo
```

The script reads your `config.json`, parses the polygon GeoJSON file, and creates a SQLite database in the `db/` folder.

Output Structure

After execution, your project directory should look like this:

```
tspo_expo/
├── 00_config.json
├── 01_config.txt
├── db/
│   └── tspo_expo.db           # SQLite database
├── json/
│   └── tspo_expo.geojson     # Input polygon file
├── tms/                      # Tile outputs (used later in the
workflow)
```

Next Steps

Once the database has been successfully created, you're ready to move forward with the TS+ workflow.

- If you have not yet created the TMS (Tile Mapping Service), your next task will be to generate it.  TSPO:TMS
- If the TMS is already complete, your next step is to open the TS+ application and begin using it.
 TSPO: App