**Tally Connector Documentation**

**Overview**

The **Tally Connector** is a Python-based application that interacts with the Tally ERP system via XML API calls. It provides a graphical user interface (GUI) using Tkinter to display active company details and ledger balances. The application also uploads ledger data to an AWS PostgreSQL database and supports both manual refresh and auto-sync modes.

**Key Features**

* **Tally API Communication:**  
  Communicates with the Tally ERP system using XML-based API requests to retrieve data.
* **Real-Time Status Monitoring:**  
  Continuously checks if Tally is running and reflects its status in the GUI (green for running, red otherwise).
* **Caching Mechanism:**  
  Caches responses for the active company and ledgers to reduce redundant API calls and improve performance.
* **Data Cleaning & Error Handling:**  
  Cleans XML responses by removing invalid characters and handles errors gracefully, both in logs and the GUI.
* **GUI Interface:**  
  Uses Tkinter to display the active company, ledger details in a table, and interactive buttons for refresh and syncing.
* **AWS PostgreSQL Integration:**  
  Connects to an AWS PostgreSQL database (using SQLAlchemy) to upload ledger records.
* **Auto-Sync Mode:**  
  Periodically checks for updates in Tally data and refreshes the GUI automatically.

**Prerequisites**

* **Python 3.x**
* **Tally ERP:** Running with the ODBC/XML interface enabled.
* **Required Python Modules:**
  + requests
  + tkinter (standard with Python)
  + xml.etree.ElementTree
  + re
  + logging
  + threading
  + datetime
  + sqlalchemy (and its PostgreSQL driver, e.g., psycopg2)

**Installation and Setup**

1. **Install Python Dependencies:**

pip install requests sqlalchemy psycopg2

(Tkinter is typically included with Python.)

1. **AWS PostgreSQL Connection:**  
   The application uses a hardcoded AWS PostgreSQL connection URL in the code.  
   *Note:* Replace the credentials and endpoint in AWS\_DB\_URL with your production values if necessary.
2. **Configuration (Optional):**
   * **Environment Variable:**  
     You can set the TALLY\_URL environment variable to specify a custom Tally server URL. The default is http://localhost:9000.

**Code Structure**

**1. TallyAPI Class**

This class is responsible for all interactions with the Tally ERP system.

* **Purpose:**
  + Sends XML requests to Tally.
  + Cleans XML responses.
  + Implements caching for the active company and ledger data.
* **Key Methods:**
  + is\_tally\_running():  
    Checks if Tally is up by sending a GET request to the server.
  + send\_request(xml\_request):  
    Sends a POST request with the given XML data to Tally (if Tally is running) and returns a cleaned XML response.
  + clean\_xml(text):  
    Cleans the XML text by removing invalid characters and ensuring proper XML structure.
  + \_generate\_request(request\_type, request\_id, fetch\_fields):  
    Dynamically creates an XML request based on the request type, ID, and fields to fetch.
  + get\_active\_company(use\_cache=True):  
    Retrieves the active company name from Tally, using a caching mechanism to avoid unnecessary API calls.
  + get\_ledgers(use\_cache=True):  
    Retrieves ledger details (names and closing balances) from Tally. Uses caching for performance.

**2. AwsDbConnector Class**

This class manages the connection to the AWS PostgreSQL database and handles uploading ledger data.

* **Purpose:**
  + Establishes a database connection using SQLAlchemy.
  + Defines and creates a ledgers table if it doesn’t already exist.
  + Inserts ledger data into the database with error handling.
* **Key Method:**
  + upload\_ledgers(company\_name, ledgers):  
    Uploads ledger records into the database within a transaction. Each record includes the company name, ledger name, closing balance, and a timestamp.

**3. TallyGUI Class**

This class provides a user-friendly Tkinter-based graphical interface for displaying Tally data.

* **Purpose:**
  + Displays the active company name and ledger data.
  + Indicates the status of Tally with a visual indicator.
  + Provides interactive buttons for refreshing data and toggling auto-sync mode.
  + Uploads ledger data to the AWS database after displaying it.
* **GUI Components:**
  + **Company Label:**  
    Shows the active company name.
  + **Status Indicator:**  
    A colored circle (green/red) that reflects Tally’s running status.
  + **Ledger Treeview:**  
    A table displaying ledger names and their closing balances.
  + **Buttons:**
    - **Refresh Button:** Triggers a manual data update.
    - **Sync Button:** Toggles the auto-sync mode.
* **Key Methods:**
  + \_setup\_gui():  
    Initializes and lays out all the GUI elements.
  + update\_status\_indicator(is\_running):  
    Updates the status indicator color based on Tally’s availability.
  + display\_error(message):  
    Shows user-friendly error messages on the GUI.
  + display\_data(company\_name, ledgers):  
    Populates the GUI with the fetched data and triggers data upload to AWS.
  + update\_gui():  
    Asynchronously fetches data from Tally and updates the GUI components.
  + toggle\_sync():  
    Enables or disables auto-sync mode.
  + sync\_gui():  
    Periodically checks for changes in the active company and updates the GUI if needed.

**4. Main Function**

* **Purpose:**
  + Initializes instances of TallyAPI and AwsDbConnector.
  + Creates the Tkinter root window.
  + Instantiates the TallyGUI class to launch the application.
* **Execution:**  
  The application starts by calling the main() function which launches the Tkinter event loop.

**Running the Application**

To start the Tally Connector GUI, run the following command:

python tally\_ledger\_viewer.py

*Ensure that the Tally ERP system is running before launching the application. The GUI will detect Tally's status, display ledger data, and upload records to the AWS database automatically.*

**Logging and Error Handling**

* **Logging:**  
  The application logs significant events and errors (such as Tally connectivity issues and database insertion errors) using Python’s logging module.
* **Error Display:**  
  Friendly error messages are displayed in the GUI if Tally is not running, not responding, or if there are issues parsing XML responses.

**Caching and Synchronization**

* **Caching:**  
  Both the active company name and ledger data are cached for a configurable duration (cache\_timeout, in seconds) to improve performance.
* **Auto-Sync:**  
  When enabled, the auto-sync feature periodically (every 2000 milliseconds by default) checks for changes in the active company and refreshes the displayed data accordingly.

**Future Enhancements**

* **Extended Data Support:**  
  Future versions may include support for additional Tally data, such as vouchers and stock items.
* **Configurable Sync Intervals:**  
  Allowing users to adjust the auto-sync interval via configuration.
* **Enhanced Error Recovery:**  
  Implementing more robust retry mechanisms and error recovery strategies.