I'll create comprehensive documentation for the Python functions that interact with the Tally XML API. This will provide information about each function, its parameters, return values, and usage examples.I'll create a comprehensive documentation file for the Python Tally XML API functions. Here's the documentation with detailed information about each function:

TallyClient Python API Documentation

This module provides a Python client for interacting with the Tally ERP/Prime XML API. It enables accessing Tally data programmatically through HTTP requests.

Table of Contents

Installation

```
pip install requests
```

Basic Usage

```
from xmlFunctions import TallyClient

# Create a Tally client
tally = TallyClient(tally_url="http://localhost", tally_port=9000)

# Test connection
if tally.test_connection():
    print("Connected to Tally")

    # Get current company
    current_company = tally.get_current_company()
    print(f"Current company: {current_company}")

    # Get list of companies
    companies = tally.get_companies_list()
    print(f"Companies: {companies}")
else:
    print("Failed to connect to Tally")
```

TallyClient Class

Constructor

```
TallyClient(tally_url="http://localhost", tally_port=9000)
```

Creates a new Tally client instance.

Parameters:

- `tally_url` (str): The URL where Tally server is running. Default: "http://localhost"
- `tally_port` (int): The port on which Tally server is listening. Default: 9000

Function Categories

Connection Methods

`test_connection()`

Tests if the Tally server is running and accessible.

Returns:

- `bool` : True if connection successful, False otherwise

`get_current_company()`

Gets the currently active company in Tally.

Returns:

- `str` : XML response containing the current company information

Collections

`get_sales_report()`

Fetches all Sales Vouchers for the current period.

Returns:

- `str` : XML response with sales vouchers

`get_companies_list(include_simple_companies=False)`

Gets a list of companies from Tally.

Parameters:

- `include_simple_companies` (bool): Whether to include simple companies in the list. Default: False

Returns:

- `str` : XML response with company list

`get_ledgers_list(company_name=None)`

Gets a list of ledgers from Tally.

Parameters:

- `company_name` (str): Company name. If provided, retrieves ledgers only for this company. Default: None

Returns:

- `str` : XML response with ledgers list

`get_stock_items_list()`

Gets a list of stock items from Tally.

Returns:

- `str` : XML response with stock items list

`get_vouchers_by_type(company_name, from_date, to_date, voucher_type="Attendance")`

Gets vouchers of a specific type.

Parameters:

- `company_name` (str): Company name
- `from_date` (str): Start date in format "01-Apr-2010"
- `to_date` (str): End date in format "04-Jun-2021"
- `voucher_type` (str): Type of voucher to retrieve. Default: "Attendance"

Returns:

- `str` : XML response with vouchers

`get_groups_list()`

Gets a list of groups from Tally.

Returns:

- `str` : XML response with groups list

Reports

`get_payslip(from_date, to_date, employee_name)`

Gets the payslip for an employee.

Parameters:

- `from_date` (str): Start date in format "YYYYMMDD"
- `to_date` (str): End date in format "YYYYMMDD"
- `employee_name` (str): Name of the employee

Returns:

- `str` : PDF data of the payslip

`get_sales_report_voucher_register(from_date, to_date, company_name, voucher_type="Sales")`

Gets the sales report using the Voucher Register.

Parameters:

- `from_date` (str): Start date in format "YYYYMMDD"
- `to_date` (str): End date in format "YYYYMMDD"
- `company_name` (str): Company name
- `voucher_type` (str): Type of voucher to include in the report. Default: "Sales"

Returns:

- `str` : XML response with sales report

`get_bill_receivables(from_date, to_date, company_name)`

Gets the bill receivables report.

Parameters:

- `from_date` (str): Start date in format "DD-MMM-YYYY"
- `to_date` (str): End date in format "DD-MMM-YYYY"
- `company_name` (str): Company name

Returns:

- `str` : XML response with bill receivables

`get_ledger_vouchers(from_date, to_date, ledger_name="Sales")`

Gets vouchers for a specific ledger.

Parameters:

- `from_date` (str): Start date
- `to_date` (str): End date
- `ledger_name` (str): Ledger name. Default: "Sales"

Returns:

- `str` : XML response with ledger vouchers

`get_group_vouchers(from_date, to_date, group_name="Sales Accounts")`

Gets vouchers for a specific group.

Parameters:

- `from_date` (str): Start date
- `to_date` (str): End date
- `group_name` (str): Group name. Default: "Sales Accounts"

Returns:

- `str` : XML response with group vouchers

`get_stock_vouchers_summary(stock_item_name, explode_vnum=True, explode_flag=False)`

Gets stock vouchers summary.

Parameters:

- `stock_item_name` (str): Stock item name
- `explode_vnum` (bool): Include voucher numbers if True. Default: True
- `explode_flag` (bool): Include detailed format if True. Default: False

Returns:

- `str` : XML response with stock vouchers summary

`get_stock_ageing(stock_group_name, from_date, to_date)`

Gets stock ageing report.

Parameters:

- `stock_group_name` (str): Stock group name
- `from_date` (str): Start date in format "DD-MMM-YYYY"
- `to_date` (str): End date in format "DD-MMM-YYYY"

Returns:

- `str` : XML response with stock ageing report

`get_list_of_accounts(from_date="", to_date="")`

Gets list of accounts.

Parameters:

- `from_date` (str): Start date in format "YYMMDD". Default: ""
- `to_date` (str): End date in format "YYMMDD". Default: ""

Returns:

- `str` : XML response with list of accounts

Objects

`get_ledger_by_name(ledger_name, from_date=None, to_date=None)`

Gets ledger by name.

Parameters:

- `ledger_name` (str): Ledger name
- `from_date` (str, optional): Start date in format "YYYYMMDD". Default: None
- `to_date` (str, optional): End date in format "YYYYMMDD". Default: None

Returns:

- `str` : XML response with ledger details

`get_voucher_by_master_id(master_id, company_name=None)`

Gets voucher by master ID.

Parameters:

- `master_id` (str): Master ID of voucher
- `company_name` (str, optional): Company name. Default: None

Returns:

- `str` : XML response with voucher details

`get_voucher_by_number_and_date(voucher_date, voucher_number, company_name=None)`

Gets voucher by number and date.

Parameters:

- `voucher_date` (str): Voucher date in format "DD-MMM-YYYY"
- `voucher_number` (str): Voucher number
- `company_name` (str, optional): Company name. Default: None

Returns:

- `str` : XML response with voucher details

`get_stock_item_by_master_id(master_id)`

Gets stock item by master ID.

Parameters:

- `master_id` (str): Master ID of stock item

Returns:

- `str`: XML response with stock item details

`get_license_info()`

Gets Tally license information.

Returns:

- `str`: XML response with license information

`parse_xml_response(xml_response)`

Parse XML response from Tally.

Parameters:

- `xml_response` (str): XML response string

Returns:

- `dict`: Parsed XML response as dictionary

Usage Examples

Retrieving Company Information

```
tally = TallyClient()
if tally.test_connection():
    # Get current company
    current_company = tally.get_current_company()
    print(f"Current company: {current_company}")

    # Get list of companies
    companies = tally.get_companies_list()
    print(f"Companies: {companies}")
```

Getting Ledger Information

```
tally = TallyClient()
if tally.test_connection():
    # Get specific ledger details
    ledger = tally.get_ledger_by_name("Cash")

    # Parse the XML response
    parsed_ledger = tally.parse_xml_response(ledger)
    print(parsed_ledger)
```

Retrieving Sales Report

```
tally = TallyClient()
if tally.test_connection():
    # Get sales report for a date range
    from_date = "20230401"  # April 1, 2023
    to_date = "20230430"    # April 30, 2023
    company_name = "My Company"

    sales_report = tally.get_sales_report_voucher_register(from_date, to_date, company_name)
    print(sales_report)
```

Getting Voucher Details

```
tally = TallyClient()
if tally.test_connection():
    # Get voucher by voucher number and date
    voucher = tally.get_voucher_by_number_and_date("01-Apr-2023", "1", "My Company")
    print(voucher)

    # Get voucher by master ID
    voucher = tally.get_voucher_by_master_id("12345", "My Company")
    print(voucher)
```

Working with Stock Items

```
tally = TallyClient()
if tally.test_connection():
    # Get list of stock items
    stock_items = tally.get_stock_items_list()
    print(stock_items)

    # Get stock ageing report
    stock_ageing = tally.get_stock_ageing("Primary", "01-Apr-2023", "30-Apr-2023")
    print(stock_ageing)

    # Get stock item details
    stock_item = tally.get_stock_item_by_master_id("6789")
    print(stock_item)
```

Error Handling

The TallyClient methods return error messages in the following format if a request fails:

- `"Error: HTTP {status_code}"` : If the HTTP request fails with a non-200 status code
- `"Error: {exception_message}"` : If an exception occurs during the request

It's recommended to check the response for these error patterns when handling the results from TallyClient methods.

```
tally = TallyClient()
response = tally.get_current_company()

if response.startswith("Error:"):
    print(f"An error occurred: {response}")
else:
    print("Successfully retrieved current company:", response)
```