**Step-by-Step Fee Schedule Process**

**Purpose:**

Updating the HRP with the monthly Medi-Cal fee schedule requires a structured and systematic approach. This document provides a detailed overview of how to convert the medical fee rates file into the HRP format. It is aimed at informing providers about the reimbursement rates for different medical services under Medi-Cal.

For this monthly Medi-Cal fee rate update, our team has developed an automated Python script. This script fetches and downloads the required data file, then performs analyses based on different criteria to generate a set of output text files. These files are provided to the HRP for the necessary updates.

**Source file Analysis:**

The source for the Fee Schedule files is part of the Medi-Cal Rates section on the California Department of Health Care Services (DHCS) website. These rates are updated every 15 days and the data related to these updates can be downloaded from this page. The "Rates" tab on this site contains detailed information about the current medical fee schedules, including various billing codes, rate descriptions, effective dates, and possibly the applicable provider types or regions.

For an in-depth analysis of the columns and data in the file, let’s examine the contents of the Excel file. Here’s a breakdown of the columns in the file:

* **Proc Type**: Indicates the type of procedure, possibly identifying different categories or groups of medical services.
* **Proc Code**: The procedure code, likely corresponding to specific medical services or treatments.
* **Procedure Description**: Describes the medical service or treatment corresponding to the procedure code.
* **Unit Value**: The assigned value for the procedure, which might be used to calculate the final rate or payment.
* **Basic Rate**: The standard reimbursement rate for the procedure.
* **Child Rate**: Specific rate applicable to pediatric cases, if different from the basic rate.
* **ER Rate**: Emergency room rate, if applicable.
* **Conv Ind**: A code indicating specific conditions or additional indicators relevant to the procedure.
* **ER Ind**: Indicates whether the procedure is associated with emergency services.
* **Cutback Ind**: Could indicate adjustments or reductions applicable under certain conditions.
* **Prof** %: Percentage applicable to professional fees, if different from the basic rate.
* **Rental Rate**: Rate applicable if the procedure involves rental of equipment or facilities.
* **Non-Physn Med Prac Ind**: Indicates whether the procedure is applicable to non-physician medical practitioners.

These columns provide detailed information about the payment and billing guidelines for various medical procedures under the Medi-Cal program. The data would be critical for healthcare providers in understanding how different services are reimbursed and any special conditions that affect these rates.

**Process Flow:**

1. **Input Data Structure** (Medi-Cal Fee Schedule): The input consists of a comprehensive Excel

spreadsheet with procedure codes and corresponding rates. These codes are categorized into "Proc Types" for classification. The fee schedule displays different rates for criteria such as Basic Rate, Child Rate, ER Rate, age, place of service, and procedure modifiers.

2. **Mapping to HRP Format**: The HRP system utilizes "Proc Types" for initial categorization. A mapping exercise has been undertaken to convert the Medi-Cal data into the format required by HRP, considering the specific criteria for each procedure type. This includes splitting multiple fees within a single Medi-Cal record into multiple rows for HRP.

3. **Output Data Structure** (HRP Config): The output files generated with mapping will be in the required format for HRP to load. The Fee Tables will align with the fee schedule and the Details will contain the code. The HRP system requires the details to accommodate the variance in fees as indicated in the Medi-Cal schedule.

4. **Process Automation**: The goal is to develop an automated and repeatable process. This involves loading the fee schedule into Python Code, interrogating the data based on the established criteria, generating data files compatible with HRP, and facilitating audits.

In practice, this process ensures that the Medi-Cal fee schedule data is accurately and efficiently ingested into the HRP system, with a keen focus on maintaining the integrity of data and supporting the subsequent updates for pricing claims. Detailed data manipulation and mapping are essential steps to translate the Medi-Cal schedule into the operational format for HRP, thereby enabling the application of appropriate fees for services rendered.

A diagram of data flow

Description automatically generated

**1. Overview of Files**

**Python Files**:

**fee\_schedule\_final.py**: This is the main script of the project that processes healthcare-related data to adjust medical fee rates based on various conditions.

**Configuration Files**:

config.json: Contains settings such as logging configuration, data columns, and other parameters.

logging\_config.ini: Configures logging settings for the Python script, such as log level and log file path.

**2. Installation and Setup:**

**System Requirements:**

Ensure Python is installed on your system. You can download it from official Python website.

**Library Installation:**

Use pip to install libraries not included in the Python Standard Library:

Ex: pip install **pandas**.

**3. Required Libraries and Modules:**

| **Module** | **Purpose** | **Installation Command** |
| --- | --- | --- |
| **zipfile** | Read and write ZIP archives | Included with Python |
| **datetime** | Manipulate dates and times | Included with Python |
| **os** | Interface with the operating system | Included with Python |
| **requests** | Send HTTP requests | **pip install requests** |
| **pandas** | Data manipulation and analysis | **pip install pandas** |
| **logging** | Logging capabilities | Included with Python |
| **json** | Parsing and outputting JSON | Included with Python |
| **shutil** | File operations like copying and archiving | Included with Python |

**4. Configuration Setup:**

config.json and logging\_config.ini should be placed in the same directory as your Python script or their paths specified in the script.

**config.json** file Contains settings for log configuration, data download URL, column names, data processing formats, and specific modifiers.

**logging\_config.ini** file Configures the logging system, including file paths, log levels, and formats.

Adjust these settings to match the requirements of the data processing tasks.

**A screen shot of a computer

Description automatically generated5. Understanding of configuration files:**

1. The config.json file contains configuration data for the Code. Here's a detailed breakdown of its contents:

* **log\_config**: Specifies the logging configuration file (logging\_config.ini).
* **download\_url**: URL for downloading Fee rates files from DHCS Website.
* **columns\_names**: Names of the columns for input Medical Rates file.
* **hrp\_columns\_fomat**: Column names in a specific format related to HRP.
* **anes\_hrp\_fomat**: Column names for anesthesia related to HRP.
* **modifier\_professional**: Code for professional services (26).
* **modifier\_technical**: Code for technical services (TC).
* **modifier\_purchase**: Code for purchased items (NU).
* **modifier\_rental**: Code for rental items (RR).
* **place\_of\_service\_code**: Code for place of service (23).
* **age\_unit\_child**: Age unit for children (3).
* **from\_age\_child**: Starting age for children (0).
* **through\_age\_child**: Ending age for children (18).

1. The logging\_config.ini file configures the logging system for the application, specifying how log messages are handled and formatted. This configuration is crucial for monitoring and debugging the application.

A screen shot of a computer

Description automatically generated

**Loggers**

* keys=root: Defines the root logger, which captures all log messages generated by the application.

**Handlers**

* keys=fileHandler,smtpHandler: Specifies two handlers:
* fileHandler: Logs messages to a file.
* smtpHandler: Sends log messages via email.

**Formatters**

* keys=simpleFormatter: Defines a single formatter for formatting log messages.

**Logger Configuration**

* root logger:
* level=DEBUG: Captures all messages at DEBUG level and higher.
* handlers=fileHandler,smtpHandler: Uses both handlers for logging.

**Handler Details**

* fileHandler:
* class=FileHandler
* level=DEBUG: Logs all messages at DEBUG level and higher.
* formatter=simpleFormatter
* args=('logging\_data.log', 'w'): Writes to logging\_data.log in write mode.

**smtpHandler:**

* class=logging.handlers.SMTPHandler
* level=ERROR: Sends emails for ERROR level messages and higher.
* formatter=simpleFormatter
* args=(('smtp.gmail.com', 587), '####@gmail.com', ['%%%%@gmail.com'], 'Error Occurred', ('####@gmail.com', 'rkzl skvo xgvo cqan'), ()): Configured to send emails via Gmail.

**Formatter**

* simpleFormatter:
* format=%(asctime)s - %(levelname)s - %(message)s: Formats log messages with timestamp, log level, and messag

**6. Detailed Documentation for fee\_schedule\_final.py**

* + 1. **Purpose:**
    2. Designed to process healthcare-related data, adjusting medical fee rates for different conditions and settings.
    3. **Core Logic:**
    4. The script processes different fee calculations based on provided data, applying specific conditions and modifiers for professional, technical, and rental rates.
    5. **Main Functionalities:**
    6. Downloading Data: Downloads datasets, handles errors, and extracts content.
    7. Data Processing Functions:
    8. Basic\_Rate\_Calculation, Child\_Rate\_Calculation, ER\_Rate\_Calculation: Calculate fees based on various conditions.
    9. Purchase\_Calculation, Rental\_Calculation: Adjust fees for purchased or rented equipment/services.
    10. anes\_calculation: Organizes anesthesia service data.

**File Output**:

* The script creates several output files based on the type of service (e.g., Surgical, Radiology, Path, and Lab, etc.
* Each file is named according to the type and includes relevant data processed by the respective functions.
* Logs information about the process flow, ensuring that all steps are tracked for debugging and verification purposes.

**Time Taken:**

****

The total time taken to execute the code to produce output for all 15 fee schedule files (along with Augment code) is 4.42 sec.

**Usage:**

To execute the script, ensure all configurations are set in config.json, and execute the Python file. The script will process the data as per the functions defined and output the respective files in the specified directory.

**Error Handling:**

The script includes error handling for file download issues, data processing errors, and logging misconfigurations, which helps in maintaining robust execution flows.

**7**. **Step by step Code Flow and Functionality:**

1. Import Required Modules:

A screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **zipfile** | For working with ZIP files. |
| **datetime** | For handling date and time. |
| **os** | For interacting with the operating system. |
| **requests** | For making HTTP requests. |
| **pandas** | For data manipulation and analysis. |
| **logging** | For logging events and errors. |
| **json** | For parsing JSON files. |
| **shutil** | For high-level file operations. |

These modules collectively enable the script to automate downloading, extracting, processing, and logging tasks efficiently.

1. Load Configurations:

A computer code with colorful text

Description automatically generated

**Action**: Opens and reads the config.json file, loading its contents into config dictionary.

**Purpose**: To retrieve configuration settings needed for script, such as the download

URL, Module parameters and column names.

1. Configure Logging

A black background with colorful text

Description automatically generated

**Action**: Configures logging using the settings from the logging configuration file

specified in config['log\_config'] and creates a logger named root.

**Purpose**: To set up logging for recording events and errors during the script's

execution.

1. Define Variables:

A computer screen shot of text

Description automatically generated

**Action**: Defines variables for the URL, current date, download filename, and

extraction path.

**Purpose**: To prepare for downloading and extracting the file with dynamically

generated names based on the current date.

1. Download and Extract File:

A screen shot of a computer program

Description automatically generated

**Action:**

* Logs the start of the download process.
* Sends an HTTP GET request to the URL.
* Writes the downloaded content to a file.
* Logs the successful download.
* Removes existing extraction directory (for. zip extract) if it exists.
* Creates a new extraction directory.
* Extracts the downloaded ZIP file to the extraction directory.
* Logs the successful extraction.

**Purpose**: To download a ZIP file from a specified URL and extract its contents to

a designated directory.

1. Process Extracted files:

A screen shot of a computer program

Description automatically generated

**Action**:

* Iterates over files in the extraction directory.
* Identifies Excel file and renames it to a standard format (**'Medi-Cal Rates {current\_date}.xlsx'**)
* Logs the renaming or existence of the file.
* Logs an error if no Excel file is found.

**Purpose**: To process extracted Excel files by renaming them and preparing them

for further data manipulation.

1. Further Data Processing:

A screen shot of a computer program

Description automatically generated

**Action**:

* Checks if a file path for the renamed Excel file exists.
* Parses the date from the filename.
* Reads the Excel file into a pandas DataFrame.
* Sets the DataFrame column names from the configuration file.
* Converts currency columns to numeric, handling errors and removing dollar signs.
* Rounds the numeric values to two decimal places for required columns.

**Purpose**: To ensure that the data in the Excel file is properly formatted and ready for analysis or further processing.

1. Basic rate calculation for necessary proc\_types:

A screen shot of a computer program

Description automatically generated

**Action:**

* Filters and creates a three DataFrame based on specific conditions from main data frame.
* Calculates FEE for the Basic, professional, and technical rates depend on conditions provided.
* Concatenates the results into a single DataFrame.
* Adds required columns like Modifier\_1, AGE\_UNIT, FROM\_AGE, etc and assign its values based on conditions
* Returns the DataFrame with the specified columns in predefined format.

**Purpose**: Calculate the basic rate for procedures, including professional and technical rates.

**Note**:

* Similar to the Basic Rate Calculation function, we have two additional functions for calculating the Child Rate(**Child\_Rate\_Calculation**) and ER Rate(**ER\_Rate\_Calculation**) for the Fee component. However, the logic for filtering the DataFrame, and the values for Modifier and other columns will differ.
* Additionally**, Purchase\_Calculation**() and **Rental\_Calculation**() functions are specifically for proc\_type='1' and are used to calculate the Fee component and other necessary modifiers and respective column fields

1. Anes Caluculation:

A screen shot of a computer program

Description automatically generated

**Action:**

* Copies the DataFrame and creates columns for anesthesia-specific data (CPT Code, CPT Descriptor, Base Unit Value).
* Adds required Modifiers (AGE\_UNIT, FROM\_AGE, etc.).
* Returns the DataFrame with the specified columns.

**Purpose:** Calculate anesthesia rates for the proc type “J” with respective logic.

**J) DataFrame Initialization and Directory Setup:**

**A screen shot of a computer program

Description automatically generated**

**Action:**

* + Creates a dictionary to hold DataFrames for each unique Proc\_Type.
  + Initializes another dictionary to hold filtered DataFrames.
  + Creates an output directory path based on the current date.
  + Ensures the output directory exists.

**Purpose**: Sets up directories and initializes dictionaries for storing DataFrames.

**k) Proc\_type logic implementation:**

**A screen shot of a computer program

Description automatically generated**

**Initial Filtering**:

* + - 1. For each Proc\_Type ('I', 'J', 'K', 'L’….,etc), the code checks if it exists in the dictionary (df\_dict).
      2. Further filters are applied based on Proc\_Code and Basic\_Rate.
      3. **Calculations**:
  1. Functions (Basic\_Rate\_Calculation, Child\_Rate\_Calculation, ER\_Rate\_Calculation, anes\_calculation) are called to calculate various rates.
  2. Each function processes the DataFrame and returns the results.

**Concatenation and Saving**:

* 1. Results are concatenated into a final DataFrame.
  2. The final DataFrame is saved as a text file with a specific naming convention based on Proc\_Type.

**Logging**:

* 1. Logs are created to indicate the successful creation of each file.

**Output file Overview:**

* + - 1. As shown in the screenshot, the Medi-Cal fee schedule, available in an MS Excel format, contains thousands of rows detailing procedure codes and corresponding rates for pricing claims. The first Column of file consists of "Proc Types" for organizational and classification purposes. Additionally, the fee schedule includes different rates to define base, child, ER, and other applicable fees.
    1. A screenshot of a computer

       Description automatically generated

Sample record and Excepted Output:

A screenshot of a computer

Description automatically generated

HRP Data format:

A close-up of a document

Description automatically generated

A document with text and numbers

Description automatically generated

**Overall code Summary:**

* **Configuration Loading and Logging Setup**:
  + Loads configurations from config.json.
  + Sets up logging based on configurations.
* **Variable Initialization**:
  + Initializes variables for URL, current date, filenames, and extraction paths.
* **File Download and Extraction**:
  + Downloads a ZIP file and extracts its contents.
* **Data Processing Functions**:
  + Defines functions for calculating various rates (Purchase\_Calculation, Rental\_Calculation, Basic\_Rate\_Calculation, Child\_Rate\_Calculation, ER\_Rate\_Calculation, anes\_calculation).
* **DataFrame Initialization and Directory Setup**:
* Initializes dictionaries to store DataFrames for each Proc\_Type.
* Creates directories for output files.
* **Filtering and Rate Calculation**:
  + Filters DataFrames based on Proc\_Type and Proc\_Code.
  + Calculates various rates and concatenates results.
  + Saves results to text files with specific naming conventions.
  + Logs the successful creation of files.

The code automates the process of downloading, extracting, processing, and saving medical procedure data into various formatted text files. It ensures thorough logging for monitoring and debugging purposes.

**Assumptions:**

* **URL Updates:** The URL used for the download button is subject to change in the future. Please ensure to verify and update the URL as necessary.

A screenshot of a computer

Description automatically generated

Right-Click on the (Download All Medi-Cal Rates) button and Select Copy Link Address for the URL. Paste the URL in Config.json.

* **Email Credentials:** The 'From' email, 'To' email, and the passkey must be updated with the credentials of the individual running the code to generate output files. Ensure these details are current and accurate.
* **Software Requirements:** It is essential to ensure that all the latest versions of Python and its corresponding libraries are installed and up to date. This guarantees compatibility and functionality of the scripts.