**Route Code Barrier Update Script Documentation**

This script is used to update route codes in a database by adding barrier tags to service points based on predefined rules. The process involves two main steps: generating updated route codes in a text file and updating the database using the generated text files. This two-step process is necessary due to the large volume of data, which makes direct updates impractical.

**Step 1: Generating Updated Route Codes in a Text File**

**Sub-step 1.1: Loading Settings from settings.txt**

The script begins by loading configuration settings from a file named settings.txt. This file contains various parameters required for the script to function correctly.

**Key Points:**

* **Loading Settings:** The script reads the settings.txt file located in the data directory.
* **Setting Variables:** It extracts variable names, types, and values, and assigns them dynamically.
* **Configuration Variables:** databaseFld and configFld are examples of configuration variables set from the file.

**Sub-step 1.2: Reading Service Points and Their Codes**

The script reads service points and their corresponding codes from a file named BarrierServicePoint.txt located in the directory specified by configFld. The list of service points near barriers can be adjusted in this file as needed.

**Sub-step 1.3: Modifying Route Codes**

The script modifies the route codes based on the service point codes read in the previous step. The function modify\_route\_code handles the logic for inserting InBarrier and OutBarrier tags based on specific conditions.

**Sub-step 1.4: Writing Updated Route Codes to a File**

The script writes the modified route codes to a text file for each database file processed. The output files are named using the pattern database name + "\_updated\_route\_codes\_barrier.txt".

**Step 2: Updating the Database with the Generated Text Files**

Due to the large volume of data, direct updates to the database are impractical. Instead, the script generates text files in Step 1 and then uses these files to update the database in Step 2.

**Sub-step 2.1: Loading Settings**

The settings loaded in Step 1 are reused in this step.

**Sub-step 2.2: Specifying Files and Table Name**

The script reads the MDB database files and the corresponding new route code files from the directories specified in the settings. It also defines the target table name (VslArrivalGen) for the database operations. The target table name can be adjusted if needed.

**Sub-step 2.3: Inserting Data into the Database**

The script connects to each .mdb file, deletes the existing data in the specified table, and inserts the new route codes from the corresponding input file. This operation is handled by the insert\_data\_into\_database function.

In summary, this script automates the process of updating route codes in database files through a two-step process: first generating updated route codes in text files, and then using these text files to update the database. This approach efficiently handles large volumes of data.