PowerShell Script Function Documentation

This document provides a detailed explanation of each function in the PowerShell script that interacts with HashiCorp Vault and MongoDB Atlas APIs, retrieves data, and logs errors into SQL Server.

# Function: Insert-ServerDataIntoSqlServer

Purpose:  
This function inserts MongoDB server data into a SQL Server table using the SqlBulkCopy method, which allows for efficient bulk data insertion.

Parameters:  
- $sqlConnectionString (String): The connection string used to connect to the SQL Server.  
- $data (Object): A data object that contains the MongoDB server details retrieved from the API.

Process:  
1. A SQL connection is established using the provided connection string.  
2. A DataTable is initialized to hold server data, with columns for attributes like hostname\_port, projectId, replicaSetName, etc.  
3. The data object is iterated over, and rows are added to the DataTable.  
4. The SqlBulkCopy class is used to efficiently insert the data into the MongoDB.MongoServers\_STG table in SQL Server.  
5. The connection is closed after the insertion is complete.

Error Handling:  
If an error occurs during data insertion, an error message is written to the console.

# Function: ConvertAndInsertPerformanceData

Purpose:  
This function processes performance data (retrieved from the MongoDB Atlas API) and inserts it into a SQL Server table using the SqlBulkCopy method.

Parameters:  
- $data (psobject): JSON data retrieved from the API.  
- $sqlServer (String): The name of the SQL Server where the data will be inserted.  
- $database (String): The database name where the performance metrics will be stored.

Process:  
1. A DataTable is initialized to hold the performance data with columns such as groupId, hostId, processId, name, etc.  
2. The performance data is parsed, and rows are added to the DataTable.  
3. The SqlBulkCopy method is used to insert the performance data into the MongoDB.PerformanceMetrics\_STG table.  
4. The SQL connection is closed after the insertion.

Error Handling:  
Errors during the insertion process are caught, and the SQL connection is closed in a finally block to ensure proper cleanup.

# Function: Get-VaultConfigAndApiKeys

Purpose:  
This function retrieves the Vault configuration and API keys from the Vault.Cert table in SQL Server and uses these details to authenticate with HashiCorp Vault.

Parameters:  
- $sqlConnectionString (String): The connection string for the SQL Server that stores Vault credentials.

Process:  
1. The Vault configuration details, such as vaultAddress, vaultNamespace, and certificate details, are fetched from SQL Server.  
2. The script authenticates with Vault using a TLS certificate.  
3. The API keys are retrieved from Vault using the authenticated token.

Error Handling:  
If an error occurs while fetching the API keys or during Vault authentication, the error is logged using the Write-ErrorLog function.

# Function: Fetch-And-Insert-MongoDBServerData

Purpose:  
This function retrieves MongoDB server data from the MongoDB Atlas API and inserts it into a SQL Server table.

Parameters:  
- $credential (PSCredential): Credentials for accessing the MongoDB Atlas API.  
- $groupId (String): The group (project) ID for which the MongoDB server data will be retrieved.  
- $sqlConnectionString (String): The connection string to the SQL Server where the data will be stored.

Process:  
1. The MongoDB server data is fetched from the Atlas API using the provided credentials.  
2. The data is inserted into SQL Server using the Insert-ServerDataIntoSqlServer function.

Error Handling:  
If any issues arise while fetching or inserting data, an error message is written to the console.

# Function: Get-ProjectIDs

Purpose:  
This function retrieves all project IDs associated with an organization from the MongoDB Atlas API.

Parameters:  
- $credential (PSCredential): Credentials for accessing the MongoDB Atlas API.

Process:  
1. The Atlas API is queried to retrieve a list of projects (groups) associated with the organization.  
2. The function returns the project IDs.

Error Handling:  
If an error occurs during the API call, the error is logged and null is returned.

# Function: Get-ServerInfo

Purpose:  
This function retrieves server information for a specific group (project) ID from the SQL Server database.

Parameters:  
- $SQLConn (SqlConnection): An open SQL connection to the database.  
- $groupId (String): The group (project) ID for which server information is being retrieved.

Process:  
1. The function queries the MongoDB.MongoServers\_STG table for servers associated with the specified groupId.  
2. The server details are returned as an array.

Error Handling:  
If the query fails, the reader is closed, and the error is propagated up the stack.

# Function: Invoke-MongoDBAtlasAPI

Purpose:  
This function invokes the MongoDB Atlas API to retrieve performance data for a specific server.

Parameters:  
- $GroupId (String): The group (project) ID.  
- $SourceServer (String): The server for which performance data is being retrieved.  
- $credential (PSCredential): Credentials for accessing the MongoDB Atlas API.

Process:  
1. The function constructs an API URL with parameters specifying the time range and granularity.  
2. The performance data is fetched and returned.

Error Handling:  
Any errors encountered during the API call are caught and logged.

# Function: Write-ErrorLog

Purpose:  
This function logs errors into a SQL Server table for monitoring and auditing.

Parameters:  
- $ErrMsg (String): The error message to be logged.  
- $TargetServer (String): The name of the server where the error occurred.  
- $CollectorName (String): The name of the process or collector that triggered the error.

Process:  
1. The error message is cleaned (to avoid SQL injection issues), and an SQL insert statement is prepared.  
2. The error is logged in the [CollectorEngine].[PowershellLoad\_Error] table.

Error Handling:  
If the error logging process fails, a secondary error message is thrown.

# Main Script Flow

Purpose:  
The main script coordinates the overall process by retrieving API keys from Vault, fetching project IDs, retrieving server and performance data from MongoDB Atlas, and logging errors as they occur.

Process:  
1. The script establishes a SQL Server connection.  
2. It retrieves Vault API keys and uses them to authenticate to MongoDB Atlas.  
3. For each project ID, MongoDB server data is fetched and inserted into the SQL Server.  
4. For each server, performance data is fetched and logged in the database.  
5. Errors encountered during the process are logged using Write-ErrorLog.