Database_Services

Service module to handle common interactions with the Linear Hash database.

Syntax

```
Response = Database_Services(@Service, @Params)
```

Returns

The meaning of the response value depends on the service.

Parameters

Parameter	Description
@Service	The name of the service being requested. **Required.**
@Params	Generic parameters. Refer to a specific service to determine the actual parameters used.

Remarks

This module provides several useful services for high level interaction with database tables and rows.

Services

Service	Description
CalculateColumn	**Usage:** `Database_Services('CalculateColumn')` **Comments:** Called directly from within a calculation column. The name of the table and column is derived from the call stack and the associated table commuter, if it exists, is called with the appropriate arguments. **Returns:** The result of the calculated column.
ClearTableHandle	**Usage:** `Database_Services('ClearTableHandle', TableName)` **Comments:** Clears the table handle array array from cache. This will force the *GetTableHandle* service to call the Open statement again. **Returns:** N/A
DeleteDataRow	**Usage:** `Database_Services('DeleteDataRow', TableName, KeyID, IgnoreSelfLock, IgnoreMFSRoutines)` **Comments:** Deletes a data row for the indicated Key ID and database table. **Returns:** N/A
GetKeyIDLock	**Usage:** `Database_Services('GetKeyIDLock', TableName, KeyID, IgnoreSelfLock)` **Comments:** Attempts to perform a semaphore lock on the indicated tablename and Key ID. **Returns:** A Boolean flag indicating if the lock request was successfully performed.

GetTableCommuter

Usage:
`Database_Services('GetTableCommuter', TableName)`

Comments:
Returns the name of the indicated table's commuter module if it exists. If it does not exist then an empty string will be returned.

Returns:
See comments.

GetTableHandle

Usage:
`Database_Services('GetTableHandle', TableName)`

Comments:
Returns an @FM list of currently attached OpenInsight database tables.

Returns:
The handle array created by the Open statement.

GetTableNames

Usage:
`Database_Services('GetTableNames', ApplicationTablesOnly, ExcludeDictionaries, ExcludeIndexes)`

Comments:
Returns an array of information related to the database table being passed in.

Returns:
See comments.

GetTableProperties

Usage:
`Database_Services('GetTableProperties', TableName)`

Comments:
Returns an array of information related to the database table being passed in.

Returns:
An @FM delimited array of table information:

Attribute	Description
<1>	Database ID
<2>	MFS/BFS list
<3>	Volume Label (if available)
<4>	Volume Path (if available)
<5>	BFS (if available)

GetUserLocks

Usage:
`Database_Services('GetUserLocks')`

Comments:
Note, this can only be done with the UD 5. This can also cause instability with the current session and may require the Task Manager to close the session.

Returns:
Returns a dynamic array of user lock information.

IsKeyIDLocked

Usage:
`Database_Services('IsKeyIDLocked', TableName, KeyID, IgnoreSelfLock)`

Comments:
Returns a Boolean flag of the lock status for the indicated table and Key ID.

Returns:
See comments.

IsKeyIDSelfLocked

Usage:
`Database_Services('IsKeyIDSelfLocked', TableName, KeyID)`

Comments:
Returns a Boolean flag of the self-lock status for the indicated table and Key ID.

Returns:
See comments

ReadDataRow	**Usage:** `Database_Services('ReadDataRow', KeyID, NotExpired, ExpirationDuration, IgnoreMFSRoutines)` **Comments:** Reads a data row for the indicated Key ID and database table. **Returns:** The requested data row.
ReleaseKeyIDLock	**Usage:** `Database_Services('ReleaseKeyIDLock', TableName, KeyID)` **Comments:** Attempts to release a semaphore lock on the indicated tablename and Key ID. **Returns:** A Boolean flag indicating if the lock release was successfully performed.
SearchIndex	**Usage:** `Database_Services('SearchIndex', TableName, ColumnName, SearchValue, UpdateIndex)` **Comments:** Returns an @FM delimited list of Key IDs that match the search value. **Returns:** See comments
SetTableAlias	**Usage:** `Database_Services('SetTableAlias', TableName, AliasName, Volume, DatabaseID)` **Comments:** Attempts to create an alias for the indicated table, volume, and database. It returns a True$ if successful or a False$ if unsuccessful. **Returns:** A Boolean flag indicating if the alias request was successfully performed.
UnlockKeyID	**Usage:** `Database_Services('UnlockKeyID', TableName, KeyID)` **Comments:** Attempts to unlock the indicated Key ID from the indicated Table Name. Note, this can only be done with the UD 5. **Returns:** A Boolean flag indicating if the unlock request was successfully performed.
VerifyLH	**Usage:** `Database_Services('VerifyLH', Tablenames, SaveList)` **Comments:** Performs a health check against the indicated tables and returns back any issues. Note: This uses the `Verify_LH` subroutine to check for GFEs. All results are stored in the SYSLHVERIFY table with a KeyID of *VolumeLabel*DatabaseID*TableName*. Returns the list of groups that have GFEs or returns an empty string if there are none. The list of GFEs or empty strings will themselves be @FM delimited to correspond with the tables passed into this service. **Returns:** An @FM delimited array of table names and associated GFE information.
VerifyLHAll	**Usage:** `Database_Services('VerifyLHAll')` **Comments:** Performs a health check against all attached tables and returns back any issues. **Returns:** Returns two lists which are delimited by an @RM. The first list is an @FM list of attached tables. The second list is an @FM list of results (groups that have GFEs or an empty string if there are none). Items in each list correspond which each other based on their list position.
WriteDataRow	**Usage:** `Database_Services('WriteDataRow', TableName, KeyID, DataRow, IgnoreSelfLock, IgnoreMFSRoutines, IgnoreAllLocks)` **Comments:** Writes a data row for the indicated Key ID and database table. **Returns:** N/A

Params

The proper use of the generic arguments are defined in the definition of each service above.