Native Tables

Command	Description
Cursors	
Activate_Save_Select subroutine	Loads a saved list of keys into cursor 0.
ClearSelect statement	Sets the list of record keys for a specified cursor to null.
Delete_Save_Select subroutine	Erases a saved list of keys.
Make.List subroutine	Creates an active select list of keys from the passed list of keys.
ReadNext, ReadNext...By statements	Reads the next row key from a selected list (cursor) into a variable. The By option allows you to read the list bi-directionally.
Reduce subroutine	Specify selection criteria for a cursor.
Rlist routine	The processing module for OpenList queries, Rlist takes the OpenList statement.
Save_Select subroutine	Saves an active select list of keys from cursor 0. The saved list can later be re-activated using Activate_Save_Select.
Select, Select... By statements	Makes each row key in a table available to the ReadNext statement in a select list. The By option allows you to access multiple select lists (cursors), and to specify sort criteria for the key list.
Dictionary Interface	Dictionary interface commands allow you to access data in tables via the dictionary from within a BASIC+ procedure.
{}	Retrieves the value of a column from the current row. The current row is defined as that stored in the variable @RECORD, with the item id stored in @ID. The dictionary being accessed must previously have been opened to the dictionary file variable @DICT.
Calculate()	Like the { } (braces) function, but can accept a variable in place of a column name. This permits a procedure to prompt for or otherwise determine which column to look up at execution time. Also requires access to @RECORD, @ID, and @DICT.
Compute_Dict()	Returns the result of a dictionary calculation. This is a shell around the Calculate() function. It opens a dictionary to @DICT, sets @ID, reads @RECORD, and calls Calculate.
Create_Symbolic routine	Creates or redefines calculated (symbolic) columns in a native table dictionary.
Dict_Depend()	Returns the column positions in a row that are required to perform a dictionary calculation.
List_Dict routine	Returns the column definitions for a specified dictionary.
File I/O	File input and output (I/O) commands allow you to read from OpenInsight data files or to write data into them.
ClearFile statement	Deletes all rows from a table but leaves the table definition.
Clear_Table subroutine	Clears the data or dictionary information from a table without deleting the table. This subroutine does not allow you to clear system tables.
Copy_OS_To_Row routine	Copies one or more operating system files to OpenEngine.
Copy_Row routine	Copies a row or a group of rows from one table to another.
Copy_Row_To_OS routine	Copies one or more OpenEngine rows to operating system files.
Delete statement	Deletes a row from an opened table.
Delete_Row routine	Deletes one or more specified rows from a table. Since Delete_Row uses the Delete statement itself, it is more efficient to use the Delete statement in a BASIC+ script. The Delete_Row routine can be called from the command line, making it useful for deleting specific records or emptying a file.

MatRead statement	Reads a row from a table into a dimensioned array.
MatWrite statement	Writes a dimensioned array to a row in a table.
Open statement	Opens a table for file I/O. Tables do not need to be closed. All I/O commands (except Xlate) use the file handle assigned by the Open statement.
Read statement	Reads a file row from a table into a variable.
Read_Column routine	Returns one or more columns from a row in a table.
ReadNext, ReadNext...By statement	Reads the next row key from a selected list (cursor) into a variable. The By option allows you to read the list bi-directionally.
ReadO statement	Read Only. Identical to Read except that the row request may be fulfilled from cache.
ReadV statement	Reads a single column into a variable from a row in a table.
Read_Row routine	Returns one or more rows from a table.
Write statement	Writes a row to a table.
Write_Column routine	Writes one or more columns to a specified row in a table.
Write_Row routine	Writes one row to a specified table.
WriteV statement	Writes a single column to a row in a table.
Xlate()	Extracts the value of a particular column in a particular row in a table.
Index	
Btree.Extract subroutine	Searches one or more Btree indexes for data matching the search criteria passed in. Returns the keys to rows having matching data.
Collect.IXVals()	Returns the list of index values for the specified indexed field in the specified file.
Create_Index subroutine	Creates a Btree, Cross Reference, or Relational index for a specified column in a table.
Delete_Index subroutine	Removes a Btree, Cross Reference, or Relational index from a specified column in a table.
Extract_SI_Keys subroutine	Searches a Btree index for a specified value and returns a list of keys based on the search value.
Get_SI_Values routine	Returns all indexed values from the index for a column.
List_Index routine	Returns information about indexes for a specified table or for all tables.
Set_Bgnd_IX_Time()	Sets the number of seconds the engine waits before and between indexing. This function is used with Set_IDXSvr() to control dedicated indexing.
Set_FSError()	Transfers @FILE.ERROR to the value returned by Get_Status().
Set_IDXSvr()	Toggles the dedicated indexing mode. To turn dedicated indexing on, pass 1; to turn dedicated indexing off, pass 0 (zero).
Update_Index subroutine	Updates or rebuilds indexes in a specified column or for all columns in an attached table.
Networking	Networking commands enable a BASIC+ procedure to set and clear row and table locks in a networking environment.
Lock statement	Sets a lock for a specified row. If the lock is held by another user, the lock fails and branches to Else logic. **Note:** A Read or Write will not check a lock. Only a lock can determine whether a lock has been set.
UnLock statement	Releases a lock set by the Lock command.

Routines	
Activate_Save_Select subroutine	Loads a saved list of keys into cursor 0.
Alias_Table subroutine	Creates a temporary synonym (an alias) for an existing table. The permanent database definition is not altered.
Attach_Table subroutine	Temporarily includes a table in the database definition. To make this permanent, run Define_Database, or use Database Manager.
Collect.IXVals()	Returns the list of index values for the specified indexed field in the specified file.
Control_Off routine	Removes transaction and domain validation controls from Native Tables.
Control_On routine	Installs transaction and domain validation controls for a table.
Copy_Row routine	Copies a row or a group of rows from one table to another.
Copy_Table subroutine	Copies a native table, the dictionary for the table, and indexing information if it exists, to a new name or location. The source and target table must be the same filing system type.
Create_Table subroutine	Creates native tables and their dictionaries.
Define_Database subroutine	Defines a database, using all currently attached tables.
Delete_Save_Select subroutine	Erases a saved list of keys.
Delete_Table subroutine	Deletes native tables and their dictionaries.
Delete_User subroutine	Removes a user from the current database. Only the administrative user can delete users.
Detach_Table subroutine	Temporarily prevents access to a single table or list of tables by removing them from the current database. The permanent database definition is not altered.
Fix_LH subroutine	Compresses overflow in a table; fixes Group Format Errors (GFEs); increments, decrements, resizes the sizelock for a table; defines a new threshold for a linear hash (OpenInsight) table.
Get_Env routine	Returns an attribute or list of attributes from the database environment for the current database.
Get.RecCount()	Returns the number of rows in a table.
Get_SI_Values routine	Returns an attribute or list of attributes from the database environment for the current database.
List_Dict routine	Returns the column definitions for a specified dictionary.
List_Index routine	Returns information about indexes for a specified table or for all tables.
List_Keys routine	Returns a list of keys from a currently attached table.
List_Tables routine	Returns the attached tables in the current database.
List_Users routine	Returns information about the users in a specified database.
List_Volume routine	Returns information about the tables in a specified volume.
Lock routine	Provides a method for coordinating access to tables, rows, or columns by setting locks.
Make.List subroutine	Creates an active select list of keys from the passed list of keys.
RowExists()	Determines whether all rows specified in a rowlist are in the specified table.

Save_Env routine	Saves the environment settings specified in the Set_Env system stored procedure, for the current database.
Save_Select subroutine	Saves an active select list of keys from cursor 0. The saved list can later be re-activated using Activate_Save_Select.
Set_Env routine	Defines the value of an environment attribute or list of attributes for the current database.
Set_MFS subroutine	Programmatically attach Modifying Filing Systems (MFS) to specified tables.
TableExists()	Determines whether the table specified exists in the current database.
Validate routine	Validates and converts data passed to the engine based on a specified validation pattern.
Verify_LH subroutine	Tests linear hash tables, and stores statistical information about data distribution and space utilization for a table or tables. Verify_LH also diagnoses group format errors (GFEs).
Write_Row routine	Writes one row to a specified table.
XREF subroutine	Reviews a string and divides it into "words," the boundaries of which are determined by delimiters you specify. XREF will exclude words according to a stop list or include words according to a go list.