DSMethod Function

Description

Executes a DataSet Object method; the DataSet Object is specified by the handle passed.

Syntax

flag = **DSMethod** *(hDS, method, arg1, arg2, arg3, arg4, arg5)*

Parameters

The DSMethod function has the following parameters:

Parameter	Description
hDS	Handle to the DataSet.
method	See *methods* below.
arg1...arg5	Method specific.

Method	Description
DS_ADDREF$	Increment DataSet reference count. For example, if you programmatically supply the handle of an existing DataSet in the DSOINSTANCE event on a form, you must increment the reference count of the DataSet to avoid it being destroyed when the form closes.
	hDS [in] The DataSet Object handle.
DS_APPEND$	Appends the default row to the end of the data set.
DS_APPEND_WORK$	Appends the workspace row to the data set.
DS_CLEAR$	Unconditionally clears the data set.
DS_COMMIT$	Commits changes made to the data set to the connected data source.
DS_COPY$	Copies the rows from one DataSet to another. The DataSets must have an identical structure: the number of columns and definition of each column must match.
	hDS [in] DataSet to copy to.
	arg1 [in] DataSet to copy from.
DS_DEFAULT_WORK$	Sets the workspace values to the data set default values.
DS_DELETE$	Deletes the current row in the data set.
DS_DESTROY$	Decrements the reference count for a DataSet and destroys the DataSet if the reference count drops to zero; if the DataSet uses a Connection Object, it releases the Connection Object (which typically will simply decrease the connection's reference count).
	hDS [in] The DataSet Object handle.
	Error returned for an invalid handle or if an error occurred destroying the object.
DS_EXECUTE$	Executes the select method and retrieves the results.
	arg1 [in] Execution type.
	arg2 [in] Retrieval type.
DS_EXTRACT_WORK$	Copies the current row in the data set to the workspace.

DS_FILTER$	Removes all rows from the current view that do not meet the specified criteria; the rows are not deleted or modified in any way and can be restored to the view using the DS_RESETFILTER$ method; the DS_FILTER$ method can be used to further filter an already-filtered DataSet as well.
	hDS [in] The DataSet Object handle.
	arg1 [in] The filter expression as it would appear in the "where" portion of a SQL script, with a few restrictions: each clause is in the form "column operator constant" where the column is the column name, the operator is "=", "!=", "<", ">", "<=", ">=", "like", "is", "is not", "in", or "not in"; "NOT" can be applied to a clause, multiple clauses can be connected with the logicals "AND" and "OR" and grouped hierarchically with parenthesis.
	arg2 [in] TRUE resets the existing filter before applying the new filter, optional, defaults to FALSE (same as using the DS_RESETFILTER$ method before using the DS_FILTER$ method).
	arg3 [in] TRUE to create a case-insensitive filter, optional, defaults to FALSE.
	arg5 [out] The BASIC+ code created to implement the filter is returned for debugging purposes.
DS_FIND$	Searches the DataSet for a row meeting the specified criteria
	hDS [in] The DataSet Object handle.
	arg1 [in] Search criteria in the same format as used for DS_FILTER$.
	arg2 [in] Find direction, optional, defaults to forwards (for more information, search for "FD_" in the insert record DSXO_API).
	arg3 [in] TRUE to create a case-insensitive filter, optional, defaults to FALSE.
	arg5 [out] The BASIC+ code created to implement the search filter is returned for debugging purposes.
DS_FINDKEY$	Using the DataSet's unique key index, DS_FINDKEY$ searches the DataSet for a row with a key matching the key in the DataSet work row. If a matching key is found, the current row is changed to the row with the matching key and success is returned. Otherwise, an error is returned.
DS_FINDNEXT$	Searches the DataSet for the next row meeting the criteria specified to DS_FIND$ and returns TRUE if a match is found.
	hDS [in] The DataSet Object handle.
	arg1 [in] Find direction, optional defaults to forward (for information , search for "FD_" in the insert record DSXO_API).
DS_FINDPREV$	Searches the DataSet for the previous row meeting the criteria specified to DS_FIND$ and returns TRUE if a match is found.
	hDS [in] The DataSet Object handle.
DS_GETERROR$	Retrieves all pending Connection Object errors from the DS/XO API.
	hDS [in] DataSet Object handle (or 0 to get API-level errors such as, DSCreate() which does not have an hDS).
	arg1 [out] Local error list (@VM-delimited).
	arg2 [out] Local severity list.
	arg3 [out] Native error list.
	arg4 [out] Native severity list.
	arg5 [out] Error text list.
	Error returned if no errors were pending.
DS_INSERT$	Inserts a row at the current position in the data set.
DS_INSERT_WORK$	Inserts the workspace row at the current position in the data set.
DS_LIMITVIEW$	Allows filtering of the current view by providing a list of row numbers to keep in the current view. This operation can be undone by useing the DS_RESETFILTER$ method.
	hDS [in] DataSet to limit the view for.
	arg1 [out] An @fm-delimited list of row numbers to keep in the current view.
DS_RESETARG$	Resets specified arguments to their default values.

	arg1 [in] Argument name/number or 0 (all).
DS_RESETDEFAULTS$	Reset DataSet column default values.
	hDS [in] The DataSet Object handle.
DS_RESETFILTER$	Removes any filter from the current view, thus making the complete view available.
	hDS [in] The DataSet Object handle.
DS_RESETSCRIPT$	Resets specified scripts to their default values.
	arg1 [in] script ID or 0 (all).
DS_ROLLBACK$	Undoes data set changes since the last commit.
DS_SORT$	Orders the DataSet based on the passed sort criteria.
	hDS [in] The DataSet Object handle.
	desc] [, colname [asc|desc]] ..."
DS_TRANSLATEFLAG$	Translates the bit-masked flag returned from the DS/XO API into TRUE$ for success and FALSE$ for failure; success includes both success and success with information (meaning possible pending messages) and failure includes an error (meaning possible pending messages), an invalid handle, and no more data.
	arg1 [in] The flag value returned from the DS/XO API
	Error returned if the DS/XO API flag value did not correspond to success.
DS_UPDATE$	Updates the database with all of the changes made to the DataSet; it is the first phase of a 2-phase DataSet commit and is repeatable; after a successful DS_UPDATE$, DS_COMMIT$ is used to resolve the changes to the DataSet itself so that the DataSet matches the state of the database (phase 2).
	hDS [in] the DataSet Object handle.
DS_UPDATE_WORK$	Copies the workspace row to the current row of the data set.

Returns

True for successful execution or **False** for failure.