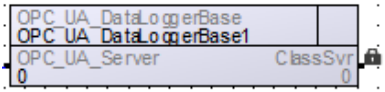
OPC_UA_DataLoggerBase

This is the base class for storing historical data of OpcUa data types. This class can only exist derived.



Using the Class

→ Serves as base class for storing OpcUa historical data.

Where the data is actually stored depends on the derivation of this class, the base class itself only provides the methods for storing and reading. The handling of the memory is done in the derivative.

→ This class also contains the interface to the C-functions "OPCUA_ParseLogValueToDataValue()" "OPCUA_ResetDataValue()" with which the byte string to be stored can be converted back into the actual OpcUa data types.

Interfaces

Server

ClassSvr	Class server Data type: DINT; DefaultInitValue: - Value range: DINT; WriteProtected: TRUE Unit: -; Retentive: FALSE

Client

OPC_UA_Server	Object channel to the OPC_UA_Server class

Global Methods

LogHistoryData	This method is called whenever a new value is to be logged from a node. The function call already contains the length and the pointer to the data, which must be stored as byte string in addition to the other transfer parameters. IN primaryKey — the index with which the data can be assigned to the node ID IN statusCode — the status of the value being stored IN sourceTime — the time from the system over which the data point was changed IN serverTime — the time from the server at which the data point was change IN valueType — the type of the data point to be stored (required for reconversion) IN dataLength — the length of the byte string to be stored IN data — the pointer to the byte string containing the data OUT: retcode — Return value 0 ... saved successfully !0 ... Error code
LogHistoryEvent	Not yet implemented (prototype)
ReadHistoryData	This method is called whenever historical data has to be read, e.g. because it is requested by a client. IN primaryKey — the index of the node ID from which the data is needed IN startTime — the start time of the requested data IN endTime — the end time of the requested data IN isInverse — indicates whether the data is needed from back to front IN numValues — a pointer to the number of data points required IN results — a pointer to the memory area where the results should be stored IN continuationPoint — a pointer to the ContinuationPoint, if it was needed in a previous query IN continuationOffset — the continuationOffset indicates how many values of the exact start time have already been read, if several exist OUT: retcode — Return value 0 … read successfully !0 … error code

Appendix

See class "OPC_UA_DataLoggerMerkerEx" for an example implementation to record data.