Overview

This is the user documentation for directly accessing the YottaDB engine without the need to go through a shim implemented in its embedded scripting language, M. A process can both call the engine directly as well as call functions written in M and exported.

Caveat: This code does not exist yet. The user documentation is being written ahead of the code, and will change in the event the code needs to differ from this documentation.

Using libyottadb

1. Install YottaDB.
2. Include the `yottadb.h` file in your C program and compile it.
3. Perform any database configuration and initialization needed (configuring global directories, creating database files, starting a Source Server process, etc.).
4. Run your program, ensuring either that `libyottadb.so` is in the load path of your program, or that it is preloaded.

Data Types

Data types are defined by including `yottadb.h` and are one of:

- User Defined Types, which in turn are one of:
  - Integer
  - Floating Point
  - Other
- Enumerated Types

User Defined Types

Integer

`ydb_int_t` and `ydb_uint_t` — Signed and unsigned integers, that are *at least* 16 bits.

`ydb_long_t` and `ydb_ulong_t` — Signed and unsigned integers, that are *at least* 32 bits.

`ydb_longlong_t` and `ydb_ulonglong_t` - Signed and unsigned integers that are *at least* 64 bits. See Numeric Considerations below.

Floating Point

`ydb_float_t` — A floating point number that is *at least* 32 bits in the representation of the underlying computing platform. See Numeric Considerations below.

`ydb_double_t` — A floating point number that is *at least* 64 bits in the representation of the underlying computing platform. See Numeric Considerations below.

Other Scalars

ydb_numeric_t — A numeric quantity in YottaDB's internal representation used to get values known to be numeric from YottaDB in order to pass them back to other functions without processing by the caller. Except when a caller needs to manipulate a numeric value returned by YottaDB, passing parameters as ydb_numeric_t types is the most efficient way to pass numeric quantities between YottaDB and C.

ydb_token_t — The type of a token that represents a value stored within YottaDB. Functions such as ydb_get() or ydb_subscript_*() used to get values — either numeric or strings — to be passed to other functions without processing by the caller can be directed to return token values of type ydb_token_t. Depending on the circumstances, using tokens may save CPU cycles on type conversion. See Tokens below.

ydb_tpfnptr_t — A pointer to a function with a single void * parameter passed by value, and a single ydb_status__t parameter passed by reference. see `Transaction Processing`_ below.

Ennumerated Types

ydb_type_t — Defines the type of value in a ydb_value_t structure. Values of a ydb_type_t are:

- YDB_CONSTSTRING_STAR — pointer to a literal string constant
- YDB_DOUBLE_STAR — pointer to a ydb_double_t value
- YDB_DOUBLE_VAL — value of type ydb_double_t
- YDB_EMPTY — the ydb_value_t structure does not contain a value
- YDB_FLOAT_STAR — pointer to a ydb_float_t value
- YDB_FLOAT_VAL — value of type ydb_float_t
- YDB_INT_STAR — pointer to a ydb_int_t value
- YDB_INT_VAL — value of type ydb_int_t
- YDB_LONG_STAR — pointer to a ydb_long_t value
- YDB_LONG_VAL — value of type ydb_long_t
- YDB_LONGLONG_STAR — pointer to a ydb_longlong_t type
- YDB_LONGLONG_VAL — value of type ydb_long_t
- YDB_NUMERIC_REQ — caller requests YottaDB to return a numeric value (i.e. one of the YDB_*_VAL types); see Numeric Considerations below
- YDB_NUMERIC_STAR — pointer to a ydb_numeric_t type
- YDB_NUMERIC_VAL — value of type ydb_numeric_t
- YDB_STRING_STAR — pointer to a structure of type ydb_string_t
- YDB_TOKEN_VAL — value of type ydb_token_t
- YDB_UINT_STAR — pointer to a ydb_uint_t type
- YDB_UINT_VAL — value of type ydb_uint_t
- YDB_ULONG_STAR — pointer to a ydb_ulong_t value
- YDB_ULONG_VAL — value of type ydb_ulong_t

Symbolic Constants

The `yottadb.h` file defines several symbolic constants, which are one of the following types:

- Function Return Codes, which in turn are one of:
  - Normal Return Codes
  - Error Return Codes
- Limits
- Other

Function Return Codes

Return codes from calls to libyottadb are of type `ydb_status_t`.

Normal Return Codes

Symbolic constants for normal return codes are prefixed with `YDB_`.

`YDB_STATUS_OK` — Normal return following successful execution.

`YDB_VALUE_EQU` — A call to a `ydb_*_compare()` function reports that the arguments are equal.

`YDB_VALUE_GT` — A call to a `ydb_*_compare()` function reports that the first argument is greater than the second (for numeric comparisons) or lexically follows the second (for string comparisons).

`YDB_VALUE_LT` — A call to a `ydb_*_compare()` function reports that the first argument is less than the second (for numeric comparisons) or lexically precedes the second (for string comparisons).

Error Return Codes

Symbolic constants for error codes returned by calls to libyottadb are prefixed with `YDB_ERR_`.

`YDB_ERR_GVUNDEF` — No value exists at a requested global variable node.

`YDB_ERR_INVMSGNNUM` — A call to `ydb_zmessage()` specified an invalid message code.

`YDB_ERR_INVSTRLEN` — A buffer provided by the caller is not long enough for the string to be returned.

`YDB_ERR_INVSUBS` — The number of entries in a `ydb_varsub_t` structure provided by the caller is insufficient for the actual number of subscripts to be returned.

`YDB_ERR_INVSVN` — A call referenced a non-existent intrinsic special variable.

`YDB_ERR_INVTOKEN` — Either a call parameter specifies that the value is a token, but the token is invalid, or libyottadb expects a token, but the tag field is not `YDB_INTERNAL`.

`YDB_ERR_LVUNDEF` — No value exists at a requested local variable node.

Limits

Symbolic constants for limits are prefixed with `YDB_MAX_`. Unless otherwise noted, symbolic constants are unsigned integers guaranteed to fit within the range of a `ydb_uint_t` type.

`YDB_MAX_IDENT` — The maximum space in bytes required to store a complete identifier (including subscripts, but not including any preceding global directory name for a global variable reference).

`YDB_MAX_MSG` — The maximum length in bytes of any message string associated with a message code. A buffer of length `YDB_MAX_MSG` bytes ensures that a call to `ydb_zmessage()` will not return a `YDB_ERR_INVSTRLEN` return code.

`YDB_MAX_STR` — The maximum length of a string (or blob) in bytes. A caller to `ydb_get()` that provides a buffer of `YDB_MAX_STR` will never get a `YDB_ERR_INVSTRLEN` error. `YDB_MAX_STR` is guaranteed to fit in a `ydb_ulong_t` type.

`YDB_MAX_SUB` — The maximum number of subscripts (keys) for a local or global variable. An array of `YDB_MAX_SUB` elements always suffices to pass subscripts.

Other

`YDB_UNTIMED` is a negative integer of type `ydb_long_t` to be provided by a caller as the timeout parameter for the functions `ydb_lock()` and `ydb_lock_incr()`.

Data Structures

Programming Notes

Numeric Considerations

The YottaDB engine internally automatically converts values between numbers and strings as needed. Thus it is legitimate to lexically compare the numbers 2 and 11, with the expected result that 11 precedes 2, and it is equally legitimate to numerically compare the strings “2” and ‘11”, with the expected result that 11 is greater than 2. The functions for numeric and lexical comparisons are different. A subscript (key) of a variable can include numbers as well as non-numeric strings, with all numeric subscripts preceding all non-numeric strings when stepping through the subscripts in order.

Furthermore, in order to ensure the accuracy of certain financial calculations, YottaDB internally stores nnumbers as, and performs arithmetic using, a scaled packed decimal representation, with optimizations for values within a certain subset of its full range of 18 significant decimal digits.

As a consequence of this:

- There are numbers which can be exactly represented in YottaDB (such as 0.1) but which cannot be exactly represented in binary floating point.
- There are numbers which are represented in 64 bit integers and binary floating point which cannot be exactly represented

Tokens