# Centrabit Script Language

Documentation for programmers

## 1. Basic information

QtPetooh is a C-like compilable programming language created specially for traders. Unlike C\C++, it is a strongly-typed programming language with no auto-casts. It is a procedural language which does not support object-oriented programming at the moment, however it supports structures like C (only predefined at the moment). Unlike C\C++, QtPetooh supports dynamic arrays with an automatic memory management. Since the language has been developed specifically for traders it has some restrictions and limitations: there are no pointers, 32bit floats and integers, chars (single symbols) as well as there is no manual memory management or garbage collectors. This language does not allow auto-casts, which means you have to cast types manually by using cast-functions - even a value *34* could not be assigned to a float variable because *34* is an integer while *34.0* is a float. These limitations allow users to write a well-controlled code.

## 2. TODO

2.1. User-defined structures.

2.2. foreach loop.

2.3. switch operator.

2.4. Timers.

2.5. File operations.

2.6. Callbacks.

2.7. Custom events.

2.8. Compiler\Binarizer.

2.9. Code encryption\decryption.

2.10. Engine code refactoring and optimization.

## 3. Types

3.1. `void` - that type means nothing, like in C.

3.2. `integer` - a 64bit signed integer variable.

3.3. `float` - a 64bit floating-point variable (`double` in C).

3.4. `string` - a Unicode string.

3.5. `boolean` - it's like a regular `bool` in C.

3.6. *Arrays* - each type (except `void`, of course) can be declared as an array of elements. For example, `integer a[5]` or `integer a[]` which means it has zero preallocated elements, it works like `std::vector` in C++.

3.7. *Structures* - they work like in C, but now users are unable to declare their own structures, they can use the predefined ones only.

3.7.1.
```
struct order
{
    integer id;
    integer marker;
    float price;
    float amount;
    string symbol;
    integer created;
    integer modified;
    boolean isAsk;
};
```

*id* - is a unique identifier that is used in private orderbook operations. Public orders have this field invalid (-1).
*marker* - is an identifier that is set by a user for his own purposes.
*symbol* - a currency pair an order is related to.
*price*, *amount* - self-explained attributes.
*created*, *modified* - a timestamp when order has been created or changed respectively.
*isAsk* - represents a type of order (to sell or to buy).

3.7.2.
```
struct transaction
{
    integer id;
    float price;
    float amount;
    float fee;
    string symbol;
```

```
    integer openTime;
    integer tradeTime;
    boolean isAsk;
};
```

*openTime* - a timestamp when an order has been placed to the orderbook.
*tradeTime* - a timestamp when the order has been executed.
*fee* - total fee of a transaction.

### 3.7.3. struct bar

```
{
    integer timestamp;
    float openPrice;
    float closePrice;
    float highPrice;
    float lowPrice;
};
```

*timestamp* - the *tradeTime* of the last transaction in a bar.
*openPrice* - a price of the first transaction in a bar.
*closePrice* - a price of the last transaction in a bar.
*highPrice* - the highest price of a transaction in a bar.
*lowPrice* - the lowest price of a transaction in a bar.

### 3.7.4. struct pair

```
{
    string key;
    string value;
};
```

3.7.5. `struct http`

```
{
    integer code;
    string body;
    boolean isHttps;
    string url;
    string host;
    integer port;
    string error;
};
```

*code* - HTTP response code that represents a status.
*body* - a body of the request/response that comes after headers.
*isHttps* - a flag that indicates whether the HTTPS is being used or not.
*url* - a string that comes after / in the URI.
*host* - a host that comes after http[s]:// part.
*port* - a port that optionally comes after the host.
*error* - an error string that is filled by the engine when errors occur.

## 4. Syntax and operators

4.1. Script declaration. Every file has to be declared with a special keyword `script`. This instruction must always be the very first line in a script.

```
script MyScript; // a name of a file must be unique within
                 // all the files included to the main script
```

4.2. Mathematical operators `+, -, *, /, %, +=, -=, *=, /=, %=, ++, --`.

4.3. Logical operators `||, &&, ==, !=, <, >, <=, >=` and `!`.

4.4. Arrays operators

```
integer a[1]; // { garbage }
a[0] = 10; // {10};
a >> 12; // append, {10,12}
a << 5; // prepend, {5,10,12}
a[1] >> 44; // insert after 2nd element, {5,10,44,12}
```

```
integer sz = sizeof(a); // returns 4 (quick operation)
integer b[] = copyof(a); // copies a to b
integer c[] = a; // c is hardlinked with a, no copy
delete a[2]; // remove 3rd element, {5,10,12}
delete a; // clear, {}
```

### 4.5. Conditions

```
import IO;

float a = 10.2;

if (a > 5.5)
    print("a is more than 5.5");
else
    print("a is less than 5.5");

if (a * 2 <= 24.0)
{
    a *= 3.0;
    print("a = " + toString(a));
}
else
{
    a += a * a;
    print("a = " + toString(a));
}
```

### 4.6. Loops

```
import IO;
integer i = 0;

while (i < 10)
{
    print("i = " + toString(i));
    i++;
}
for (integer j = 0; j < 10; ++j)
```

```
{
    print("j = " + toString(j));
}

do
{
    print("i = " + toString(i));
} while (i >= 0);
```

### 4.7. Functions

```
import IO;

// forward declaration is not supported
string multiplyContent(integer[] numbers, integer count)
{
    string result;

    for (integer i = 0; i < sizeof(numbers); ++i)
    {
        string symbol = toString(numbers[i]);
        string part;

        for (integer j = 0; j < count; ++j)
            part += symbol;
        result += part;
    }

    return result;
}

integer nums[4];

nums[0] = 1;
nums[1] = 2;
nums[2] = 3;
nums[3] = 4;
```

```
print(multiplyContent(numbers, 2)); // prints 11223344
```

## 4.8. Events

- onBalanceChanged(string exchange, string symbol, float amount) - event for any balance value changed
- onLastPriceChanged(string exchange, string symbol, float amount) - event for any buy/sell transaction amount
- onAskPriceChanged(string exchange, string symbol, float amount) - event for the best order book ask price changed
- onBidPriceChanged(string exchange, string symbol, float amount) - event for the best order book bid price changed
- onOwnOrderOpen(string exchange, order o) - result of buy/sell command
- onOwnOrderModified(string exchange, order o) - event for order modification
- onOwnOrderFilled(string exchange, transaction t) - event when order filled completely
- onOwnOrderError(string exchange,
  string symbol,
  integer marker,
  string error) - event for orders failed to open
- onPubOrderOpen(string exchange, order o) - new order appeared in public orderbook
- onPubOrderModified(string exchange, order o) - order modified in public orderbook
- onPubOrderFilled(string exchange, transaction t) - new public transaction

```
/*
  Events are predefined callback-functions that are being
     called by Script Engine.
  A user has to implement a body of these callbacks to be able
    to use them.
*/
```

```
import IO;


event onBalanceChanged(string exchange,
                       string symbol,
                       float amount)
{
    print("You have " + symbol + " = " + toString(amount) +
          " remaining on " + exchange);
}
```

### 4.9. Import

Imports work like #include in C\C++.

Also a user can import predefined libraries:

- IO
- Math
- Strings
- Trades
- Time
- Charts

```
import IO; // allows to use print function
import "mylib.csh"; // imports a user-written code. Must have csh-suffix
```

## 5. Built-in libraries and functions

### 5.1. Built-in functions

```
integer toInteger(any);
boolean toBoolean(any);
float toFloat(any);
string toString(any);
integer sizeof(any[]);
integer indexof(any, any[]);
any[] copyof(any[]);
integer random(integer from, integer to);
string getEnv(string varName);
```

```
void sleep(integer timeout);
void msleep(integer timeout);
```

### 5.2. IO

```
void print(any);
```

### 5.3. Math

```
integer abs(integer value)
float fabs(float value)
float sqrt(float value)
float pow(float value, float power)
integer floor(float value)
float round(float value)
float sin(float value)
float cos(float value)
float tan(float value)
integer min(integer a, integer b)
integer max(integer a, integer b)
float fmin(float a, float b)
float fmax(float a, float b)
```

### 5.4. Strings

```
integer strlength(string str)
string substring(string str, integer pos, integer len)
integer strfind(string base, string what)
void strreplace(string base, string before, string after)
void strremove(string str, integer pos, integer len)
void strinsert(string base, integer pos, string str)
```

### 5.5. Trades

```
float getTotalBalance(string exchange,
                      string currency)
float getAvailableBalance(string exchange,
                          string currency)
float getLockedBalance(string exchange,
                       string currency)
order getOwnOrderById(string exchange,
                      string symbol,
                      integer orderID)
```

```
order[] getOwnOrders(string exchange,
                     string symbol)

order[] getOwnOrdersAsks(string exchange,
                         string symbol)

order[] getOwnOrdersBids(string exchange,
                         string symbol)

order[] getOrderBookByRange(string exchange,
                            string symbol,
                            float priceLower,
                            float priceUpper)

order[] getOrderBookByRangeAsks(string exchange,
                                string symbol,
                                float priceLower,
                                float priceUpper)

order[] getOrderBookByRangeBids(string exchange,
                                string symbol,
                                float priceLower,
                                float priceUpper)

transaction[] getOwnTrades(string exchange,
                           string symbol,
                           integer timeStart,
                           integer timeEnd)

transaction[] getPubTrades(string exchange,
                           string symbol,
                           integer timeStart,
                           integer timeEnd)

float getPriceOfAmount(string exchange,
                       string symbol,
                       boolean isAsk,
                       float amount)

float getAmountOfPrice(string exchange,
                       string symbol,
                       boolean isAsk,
                       float price)

float getOrderBookAsk(string exchange,
                      string symbol)

float getOrderBookBid(string exchange,
                      string symbol)

order getPubOrder(string exchange,
                  string symbol,
                  float price)

integer buy(string exchange,
            string symbol,
            float amount,
            float price,
            integer marker)

integer sell(string exchange,
             string symbol,
```

```
                float amount,
                float price,
                integer marker)

integer buyMarket(string exchange,
                  string symbol,
                  float amount,
                  integer marker)

integer sellMarket(string exchange,
                   string symbol,
                   float amount,
                   integer marker)
```

## 5.6. Time

Time that is represented in integer is a number of **microseconds** since epoch. For example, *Friday, December 24, 2021 12:34:17 AM* is *1640306057000000* microseconds since epoch approximately.

```
string timeToString(integer timestamp, string format)
integer stringToTime(string timestamp, string format)
integer getCurrentTime()
```

Format expressions:

| Expression | Output |
|---|---|
| h | The hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display) |
| hh | The hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display) |
| H | The hour without a leading zero (0 to 23, even with AM/PM display) |
| HH | The hour with a leading zero (00 to 23, even with AM/PM display) |
| m | The minute without a leading zero (0 to 59) |
| mm | The minute with a leading zero (00 to 59) |
| s | The whole second, without any leading zero (0 to 59) |
| ss | The whole second, with a leading zero where applicable (00 to 59) |
| z | The fractional part of the second, to go after a decimal point, without trailing zeroes (0 to 999). |

| | |
|---|---|
| | Thus "s.z" reports the seconds to full available (millisecond) precision without trailing zeroes |
| zzz | The fractional part of the second, to millisecond precision, including trailing zeroes where applicable (000 to 999) |
| AP or A | Use AM/PM display. |
| ap or a | Use am/pm display |
| t | The timezone (for example "CEST") |

| | |
|---|---|
| d | The day as a number without a leading zero (1 to 31) |
| dd | The day as a number with a leading zero (01 to 31) |
| ddd | The abbreviated localized day name (e.g. 'Mon' to 'Sun'). |
| dddd | The long localized day name (e.g. 'Monday' to 'Sunday'). |
| M | The month as a number without a leading zero (1 to 12) |
| MM | The month as a number with a leading zero (01 to 12) |
| MMM | The abbreviated localized month name (e.g. 'Jan' to 'Dec'). |
| MMMM | The long localized month name (e.g. 'January' to 'December'). |
| yy | The year as a two digit number (00 to 99) |
| yyyy | The year as a four digit number, possibly plus a leading minus sign for negative years. |

## 5.7. Charts

```
void setChartsExchange(string exchange)

void setChartsSymbol(string symbol)

void setLineName(string name,
                 string exchange,
                 string symbol)
```

```
void setLinePosition(string position,
                     string exchange,
                     string symbol)

void setLineColor(string color,
                  string exchange,
                  string symbol)

void drawLine(integer time,
              integer price,
              string exchange,
              string symbol)

void drawPoint(integer time,
               integer price,
               integer isAsk,
               string comment,
               string position,
               string exchange,
               string symbol)

void clearCharts(string exchange,
                 string symbol)
```

## 5.8. HTTP

This library implements a basic functionality of HTTP protocol. One can use it to request third party services in JSON, url-encoded or any other format **text** format (binary data in bodies is not supported yet). *Please note*: you have to use functions presented below to construct `http` objects. If the `http` object is constructed manually, the system will not execute such a request. See description of `http` structure above to understand how to access data.

```
http createHTTP(string uri, pair[] headers)
string getHeader(http response, string header)
string makeUrlEncodedBody(pair[] body)
pair urlEncode(string key, string value)
pair[] urlDecode(string body)
void requestGET(http request)
void requestPOST(http request)
void setHTTPProxy(string user,
                  string password,
                  string host,
                  integer port)
```