VIALATM SCRIPT

VERSION 2.3

Table of contents

INTRODUCTION

A script is a sequence of statements that are executed when new messages arrive from the object.

Each operator is located on a separate line.

The lines starting with the "#" character are comments.

Operator and variable identifiers must be entered in upper case.

Variable IDs cannot exceed 16 characters.

As variables, you can use the current attributes of an object or an incoming message. Variables are stored in the attributes of the object.

BASIC OPERATORS

LET – ASSIGNMENT OPERATOR

To assign values to variables, use the LET statement

LET ALFA = 134

LET BETA = ARG * 1.3

The right-hand side can be the function

LET GAMMA = @ROUND (ALFA * 1.3)

The right side can contain only one action. If you need to perform a more complex calculation, for example, GAMMA = (ARG + 10) * 22, then it must be split into several operators:

LET GAMMA = ARG + 10

LET GAMMA = GAMMA * 22

If necessary, you can use hexadecimal constants. For them, you need to add the prefix 0x:

LET OMEGA = 0xF1AB

INIT – INITIALIZATION OPERATOR

The initialization operator acts as a LET statement. But if the variable is already defined and has a value, then an assignment ignored.

INIT ALFA = 134

In fact, this is a shortened form of the statement:

IF @NOTEXIST ALFA THEN LET ALFA = 1234

IF – CONDITIONAL OPERATOR

To perform different computations or actions depending some condition IF operator is used:

IF <condition> THEN <operator1>

IF <condition> THEN <operator1> ELSE <operator2>

In the first case, if the condition is true, operator1 is executed.

In the second case, if the condition is true, operator1 is executed, otherwise operator2 is executed.

Examples of conditional statements:

IF ATTR1>10 THEN LET ATTR2=22

IF ATTR2<= ATTR1 THEN LET ATTR2=22 ELSE RETURN

Comparison conditions:

== is equal to

!= is not equal

>= greater than or equal to

<= less than or equal to

> greater than

< less than

&& bitwise conjunction of operands. The result is true if, as a result of conjunction, the result is greater than 0

|| bitwise disjunction of operands. The result is true if, as a result of disjunction, the result is greater than 0

A function can be a condition. If the function returns a result other than zero, then the condition is true, otherwise false. Example:

IF @GETBIT ATTR1 3 THEN LET ATTR2 = 101 ELSE LET ATTR2 =102

DEF – VISUALIZATION OF A VARIABLE

To be able to visualize a variable in the web application interface, the DEF operator is used.

DEF ATTR1 : "Sensor 1"

DEF ADC1 : "External power"

After defining this statement in the script, the selected attribute appears in the list of attributes for the tooltips and can be selected for visualization.

With this operator, you can change the name of the predefined attributes of the object. For example, for speed:

DEF V : "Instantaneous speed"

LABEL – DEFINE A LABEL

To define a label, you must enter a string with an identifier ending with ":". By using the JUMP operator, you can go to the specified label.

MYLABEL:

JUMP – JUMP TO THE LABEL

The JUMP statement allows you to jump to the specified label. If there is no given label, the script ends. Search for a label goes only down the script.

JUMP MYLABEL

RETURN – EXIT FROM THE SCRIPT

The RETURN statement allows you to exit the script processing program. As a rule, the operator is used in conditional statements.

IF X2>1 THEN RETURN ELSE LET DIST=102

EVENT – ADDING AN EVENT

The EVENT statement allows you to add a new event in the incoming message.

EVENT "Dangerous braking"

As a rule, the operator is used in conditional statements.

IF X5 == 1 THEN EVENT "Dangerous braking"

You can add several different events to the message.

To add an SOS event, there is a special version of the operator - SOS:

IF X2 == 1 THEN SOS

This is an analogue of the EVENT "SOS"

CLR_EVENTS – CLEARING EVENTS IN A MESSAGE

Sometimes it is necessary to delete events in the received message. The CLR_EVENTS statement allows you to delete all events in the incoming message.

CLR_EVENTS

CLR – DELETING MESSAGE ATTRIBUTES

The CLR operator allows you to remove attributes from incoming messages.

CLR X1,INPUT,D2,D

Removing unused attributes allows you to optimize the volume of stored messages. At the same time, the efficiency of working with archive data is significantly increased.

REMOVE – CLEARING OBJECT ATTRIBUTES

The REMOVE statement allows you to delete the attributes of an object.

REMOVE OUTPUT,INPUT,IO2,ODOM,IO1,X2

It is forbidden to delete the attribute DT - the time the last message was received. An exception is thrown when trying to do this.

SYN – DEFINITION OF A SYNONYM FOR AN ATTRIBUTE

The SYN statement allows you to define a synonym for an attribute. This is necessary when the attribute has different identifiers in different objects, but it is necessary to compare their data on one graph.

SYN COMMONATTR : ATTROBJ_N

By defining in different objects operators:

DEF COMMONATTR : "Attribute A"

SYN COMMONATTR : ATTROBJ_1

and

DEF COMMONATTR : "Attribute A"

SYN COMMONATTR : ATTROBJ_2

You can display "Attribute A" on one chart, which has different identifiers in these objects.

DEL_EVENT – DELETING AN EVENT FROM A MESSAGE

The operator allows you to delete an event from an incoming message.

Example:

Let's say the incoming message contains events: low battery; stockade. After the statement is executed:

DEL_EVENT "stockade"

the incoming message will contain only the low battery event.

REPLACE_EVENT – RENAME AN EVENT IN A MESSAGE

The operator allows you to rename the event in the incoming message.

Example:

Let's say the incoming message contains events: low battery; stockade. After the statement is executed:

REPLACE_EVENT "stockade" "leave zone"

the incoming message will contain events: low battery; leave zone.

COMMAND – SENDING A COMMAND VIA GPRS

The operator allows you to send a command in response to an incoming message.

Example:

Suppose you need to set the positioning frequency in a certain zone (1234) with a frequency of 60 seconds, and outside - 10 seconds. This can be solved by the following script (for example, trackers working with the Coban protocol):

```
LET CURRENT_ZONE = @INZONE 1234
IF @NOTEXIST SAVED_ZONE THEN LET SAVED_ZONE = CURRENT_ZONE
IF CURRENT_ZONE == SAVED_ZONE THEN RETURN
IF CURRENT_ZONE == 1 THEN COMMAND "**,imei:%IMEI,C,60s" ELSE COMMAND "**,imei:%IMEI,C,10s"
LET SAVED_ZONE = CURRENT_ZONE
```

OPERATORS TO CHANGE THE ICON OF THE OBJECT

ICON_COLOR – SETTING ICON COLOR

To change the icon color, use the ICON_COLOR statement. The operator applies only to icons in the format GIF and PNG. This only changes the points that are black in the original image.

```
ICON_COLOR = "442233"
```

The string on the right side of the expression specifies a color in the format RRGGBB

You can use the preset colors:

```
ICON_COLOR = TEAL
```

As a rule, the operator is used in conditional expressions:

```
IF V>70 THEN ICON_COLOR = "996633" ELSE ICON_COLOR = GREEN
```

WHITE	SILVER	GRAY	BLACK
RED	MAROON	YELLOW	OLIVE

LIME	GREEN	AQUA	TEAL
BLUE	NAVY	FUCHSIA	PURPLE

ICON_SIZE – SETTING THE SIZE OF THE ICON

To change the size of the icon (in pixels), use the ICON_SIZE operator:

ICON_SIZE = 24

The right side of the expression must be a numeric constant in the range from 16 to 256

ICON_IMG – SETTING ICON IMAGE

To change the icon image, use the ICON_IMG statement:

ICON_IMG = "6088340.png"

The right part - specifies the name of the icon from the set of custom icons (the search is iperformed among the icons downloaded by the owner of the device, and not by the current user!).

If the right-hand side contains the group prefix, then the icon is downloaded from the group of public icons:

ICON_IMG = "car/car7.png"

In the service there are groups: boat, bus, car, goods, moto, phone, plan, specs, user.

The images that correspond to the files can be viewed in the client's web interface.

FUNCTIONS

All functions start with "@"

@ROUND - ROUNDING

The function rounds the value of the variable to the specified decimal point. Examples:

LET UA = @ROUND 124.234567 2

result - UA will be equal to 124.23

LET UA = @ROUND UA 0

variable UA will be equal to 124 (0 - rounding to the integer)

@GETBIT – GETTING THE VALUE OF A GIVEN BIT

The function returns a value of 0 or 1, depending on the value of the corresponding bit in the variable. Bits are numbered from the low-order bit. Numbering starts from 1. Examples:

@GETBIT X7 1 : with the value of variable 5(0000010**1**), the function returns 1; at the value of 4(0000010**0**) — 0

@GETBIT X5 4 : with the value of variable 12 (0000**1**100) the function returns 1; at the value of 7(0000**0**111) - 0

@GETBITHEX – GETTING THE VALUE OF A GIVEN BIT FROM HEX STRING

The function returns a value of 0 or 1, depending on the value of the corresponding bit in the variable represented as a hex string. Bits are numbered from the low-order bit. Numbering starts from 1. Examples:

@GETBITHEX X7 1 : with the value of variable «05»(0000010**1**), the function returns 1; at the value of 4(0000010**0**) — 0

@GETBITHEX X5 4 : with the value of variable «0C» (0000**1**100) the function returns 1; at the value of 7(0000**0**111) - 0

@GETVALHEX – GETTING THE VALUE FROM HEX STRING

The function returns a value, depending on the values of the corresponding **bits** in the variable represented as a hex string. Bits are numbered from the low-order bit. Numbering starts from 1. The first parameter specifies the name of the variable, the second - starting bit from which the numerical value will be formed, the last argument specifies the number of bits to be selected. Examples:

@GETVALHEX X7 5 3 : with the value of variable «05»(000**001**01), the function returns 1; at the value of «14»(000**101**00) — 5

@GETVALHEX X5 4 2 : with the value of variable «0C» (0000**11**00) the function returns 3; at the value of «AC07»(101011000000**01**11) - 1

@NOTEXIST – CHECKING A VARIABLE

The function returns a value of 0 if the specified variable exists (is defined) and 1 otherwise. As a rule, the operator is used in conditional statements.

IF @NOTEXIST ODOM THEN LET ODOM = MILEAGE ELSE LET ODOM = ODOM+MILEAGE

@DIST – DISTANCE TO A GIVEN GEO LOCATION

The function returns the distance from the geo-coordinate received in the current message from the object to the coordinate specified in the parameter (if the current message does not contain coordinates, then the last received coordinate of the object is taken). The result is in kilometers.

LET DIST1 = @DIST 41.39296,2.13922

If the coordinates in the current message were - 41.39296.2.13822, the variable DIST1 will be set to 0.083

@DSCALE – SELECTING A VALUE BY A DISCRETE SCALE

The function returns a string value from a discrete scale based on the value of the parameter:

@DSCALE PARAM "Str1",V1,"Str2",V2,"Str3"

If the value of PARAM <V1, returns Str1; if the value of PARAM <V2, returns Str2 otherwise Str3 is returned. Example:

LET SPEEDMODE = @DSCALE V "Normal",60,"Warning",90,"Alert"

The variable SPEEDMODE stores the text about speed mode.

@FSCALE – SELECTION BY A LINEAR SCALE

The function returns a numeric value from the linear function:

@FSCALE PARAM x1,y1,x2,y2

Two points (x1, y1) and (x2, y2) define the linear function ax + b

The PARAM parameter is the argument of the function.

Example:

LET FUEL = @FSCALE X1 0,0,1000,2000

It is convenient to use the function to determine the fuel in the tank by the value of the sensor (a linear function in this case sets the calibration table).

@TOSIGNED – CONVERSION TO SIGNED VALUES

The values coming from some types of devices (for example TELTONIKA) are treated as positive integers. But the values can be transmitted as integers (positive or negative). In such cases, for attributes, you need to apply the function:

@TOSIGNED ATTR 1

The first parameter specifies the attribute identifier, the second - size in bytes (1,2 or 4).

Example:

Suppose the SNUM parameter was set to 136. After the statement is executed:

LET SNUM = @TOSIGNED SNUM 1

SNUM will be -120

@HASEVENT – EVENT CHECK

The function checks the presence of a specific event in the incoming message. If there is an event in the message, it returns 1 and 0 otherwise.

Example:

Let's say the incoming message contains events: low battery; door alarm. After the statement is executed:

IF @HASEVENT "door alarm" THEN LET DOOR_OPEN_COUNTER = DOOR_OPEN_COUNTER + 1

the DOOR_OPEN_COUNTER variable will be incremented.

@INZONE – PRESENCE CHECK IN GEOFENCE

The function checks the presence of an object in the specified geofrnce. If the object is in the zone, then 1 is returned, and otherwise 0. The geofence is specified by the identifier.

Example:

LET MYZONE = @INZONE 1234

The MYZONE variable will be 1 if the object is in the geofence with ID 1234 and 0 otherwise.

@STAYINZONE – TIME OF STAYING IN ZONE

The function returns the time of staying in the specified zone (in seconds). If 0 is returned, then either the object is out of the zone, or entered in the specified zone in the current event.

Example:

LET PZONE = @STAYINZONE 1234

SPECIAL VARIABLES

In the scripting language, there are special variables that simplify the script writing. These variables begin with the symbol "!" . Variables are read-only and can be used in expressions.

!DIST - DISTANCE

The variable contains the distance from the previous geo-coordinate of the object.

LET TOT_DIST = TOT_DIST + !DIST

!TIME - TIME INTERVAL

The variable contains the time interval (in seconds) between the current and last message.

LET MOV_TIME = MOV_TIME + !TIME

!ACCEL - ACCELERATION

The variable contains the acceleration calculated as the difference between the speed in the incoming message and the speed in the previous message divided by the time interval between events. The unit of measurement is meters per second squared.

LET ACV = !ACCEL

MACROSES

In the scripting language, there are macros that simplify script writing. Macros begin with "."

.COLOR_FOR_SPEED – SETTING COLOR ICONS FOR SPEED

The macro sets the coloring of the object icon depending on the current speed.

Example:

.COLOR_FOR_SPEED GREEN,60,"0000FF",90,RED

At a speed of less than 60 km / h, the icon will be green, from 60 to 90 - blue and at a speed of 90 or higher - red. Gradation and coloring can be expanded.

.ICON_COLOR – SETTING COLOR ICONS BY PARAM VALUE

The macro sets the coloring of the object icon depending on the given param value.

Example:

.ICON_COLOR V 0 GREY; 60 0000FF; 90 RED; * BLACK

If speed (V) less or equal 0 icon color will be grey; at a speed of equal or less than 60 km / h, the icon will be blue, from 60 to 90 - red and at a speed of higher than 90(*) color will be black.

.FUEL_BY_SENSOR – FUEL BY SENSOR

The macro sets the current volume of fuel by the sensor value. Also, a FUEL attribute is created that stores the current volume of fuel. The attribute data can be displayed in the object's tooltips.

Example:

.FUEL_BY_SENSOR X8,0,0,1000,40

Based on the parameters 0,0,1000,40 and the value of the sensor X8, the current volume of fuel is set in FUEL attribute.

.ACCELERATION – ACCELERATION

The macro sets the ACCEL attribute, which stores the current acceleration. The attribute data can be displayed in the object's tooltips.

Example:

.ACCELERATION

.ODOMETER – ODOMETER

The macro sets the ODOMETR attribute that stores the current mileage of the object. The attribute data can be displayed in the object's tooltips.

Example:

.ODOMETER

When the mileage exceeds 100,000 km, the mileage is automatically updated to zero.

.REMOTENESS – REMOTENESS FROM A GIVEN GEO-COORDINATE

The macro sets the remoteness of the object from the specified geo-coordinate.

Example:

.REMOTENESS D_PLACE1,"Distance to Base",48.14045,11.56754

The first parameter D_PLACE1 specifies the attribute in which the remoteness is stored.

The second parameter specifies the name under which remoteness can be displayed in the tooltips.

The third and fourth parameters specify the latitude and longitude of the position to which the distance is calculated.

.SOS_SENSOR – GENERATING THE SOS EVENT BY SENSOR

The macro allows you to generate an SOS event, depending on the value of the specific bit in the specified message sensor/attribute.

Example:

.SOS_SENSOR INPUT,2

The first parameter specifies the attribute in which the corresponding bit is checked.

The second parameter specifies the bit to be checked.

If the bit is 1, then an SOS event is generated.

.IGNITION_SENSOR – IGNITION STATUS BY SENSOR VALUE

The macro allows you to set the Ignition / Engine On / Off (SS) attribute, depending on the specific bit value in the specified sensor.

Example:

.IGNITION_SENSOR INPUT,3

The first parameter specifies the attribute in which the corresponding bit is checked.

The second parameter specifies the bit to be checked.

If the bit is 1, the ignition attribute(SS) is set to 1, otherwise to 0

.AVERAGE_SPEED – AVERAGE SPEED

The macro sets the average speed of the object. An AV attribute storing the average speed is created. The attribute data can be displayed in the object's tooltips.

Example:

.AVERAGE_SPEED

EXAMPLES

```
LET V = V * 0.62137  # velocity in miles/hour
IF V>90 THEN EVENT "Over speed"
DEF TP : "Tire pressure"
LET TP = IO181 * 0.001
```

```
IF V>100 THEN ICON_COLOR = "996633" ELSE ICON_COLOR = GRAY
DEF TOT_ODOM : "Odometer"
IF @NOTEXIST TOT_ODOM THEN LET TOT_ODOM = 0
IF @NOTEXIST ODOM THEN LET ODOM = 0
LET ODOM = ODOM + !DIST
IF ODOM>30000 THEN EVENT "Inspection required"
LET TOT_ODOM = TOT_ODOM + !DIST
```