К-СОФТ ИНЖИНИРИНГ

# KSE Platform 3.4

## Руководство разработчика: Описание API Lua серверных объектов

г. Санкт-Петербург

# Содержание

# 1. О документе

1. Настоящий документ предназначен для квалифицированных специалистов, обладающих базовыми и продвинутыми знаниями в области программирования.
2. Документ описывает специализированные функции и свойства, доступные в серверных программах. Функции серверных программ предоставляют следующие возможности:

   • создание, изменение, удаление агентов, привязок, каталогов и тегов;

   • организация тревог и событий;

   • запуск, остановка агентов и привязок;

   • подготовка схем данных и генерация отчетов;

   • чтение и запись значений в устройства по протоколу modbus;

   • получение данных с веб-сервисов.
3. ООО "К-СОФТ Инжиниринг" оставляет за собой право на внесение изменений в настоящий документ в любое время. Если изменения будут носить масштабный характер, например: обновление дизайна, создание нового документа, существенные изменения сути, то они будут зафиксированы в очередном ReleaseNotes.
4. Вопросы по документу, а также запросы на техническую поддержку ПО можно отправить по адресу: support@k-soft-spb.ru.

**ВАЖНО!**

Внутренние пользователи ПО оформляют запросы в bitrix. Внешние - любым доступным способом (мессенджеры, электронная почта и т.д.)

# 2. Соглашения и условные обозначения, принятые в документе

В настоящем документе используются:

- Соглашения:

| | |
|---|---|
| Меню, названия диалоговых окон и их свойства, названия документов, ключевые слова. | **Жирный шрифт** |
| Команды, примеры программ. | `Runtime.exe` |
| Имена файлов и пути. | *Курсив* |
| Ссылка на раздел настоящего документа (в скобках указан номер страницы). | *ссылка* |

- Условные обозначения:

| | |
|---|---|
|  | Информация обязательная для прочтения/выполнения. |
|  | Отсылка к документу, который может содержать более полное описание изучаемой темы. |

# 3. Управление алармами

Аларм (тревога) представляет собой объект типа LuaAlarm. Источником тревоги может быть каталог, тег, агент, привязка.

## 3.1. Свойства

| **Имя** | **Тип** | **Описание** |
|---|---|---|
| *Acked* | Boolean | Возвращает **true**, если аларм квитирован, иначе возвращает **false**. |
| *AckedBy* | String | Возвращает отображаемое имя пользователя, который квитировал аларм. Если аларм не квитирован, ничего не возвращает. |
| *Comment* | String | Возвращает комментарий квитирования. Если аларм не квитирован, ничего не возвращает. |
| *Enabled* | Boolean | Возвращает **true**, если аларм активен (включен), иначе - **false**. |
| *Severity* | Number | Возвращает приоритет аларма. |
| *Message* | String | Возвращает полный текст сообщения аларма. |
| *ReceiveTime* | DateTime | Возвращает время возникновения тревоги |
| *SourceName* | String | Возвращает имя источника тревоги, заданное методом Enable() |

## Acked

**Синтаксис:**

```
flag = alarm.Acked
```

**Результат:**

Возвращает **true**, если тревога квитирована, иначе возвращает **false**.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('instance')
if not alarm.Acked then
local err = alarm:Ack('acking...')
if err ~= nil then print(err)
end
end
```

В примере получаем тревогу с именем `'instance'` в переменную `alarm`. Если тревога не квитирована (`not alarm.Acked`) производим квитирование с сообщением `'acking...'`. В случае ошибки выводим сообщение об ошибке на печать.

## AckedBy

**Синтаксис:**

```
string = alarm.AckedBy
```

**Результат:**

Возвращает в виде строки отображаемое имя пользователя, который квитировал тревогу. Если тревога не квитирована, ничего не возвращает.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('alarm')
if alarm.Acked then if alarm.AckedBy ~= 'Администратор' then
local err = alarm:Enable('test', 500,'Enabling...')
if err ~= nil then print(err) end
end
end
```

В примере получаем тревогу для источника в переменной root (корневой каталог Tags), проверяем её состояние квитирования, если тревога квитирована любым пользователем, кроме администратора, то активируем ее повторно.

## Comment

**Синтаксис:**

```
text = alarm.Comment
```

**Результат:**

Возвращает комментарий квитирования. Если тревога не квитирована, ничего не возвращает.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
if root ~= nil then
local alarm = root:GetAlarm('Test')
if alarm.Acked then
print(alarm.Comment)
end
end
```

В примере получаем тревогу, если она квитирована, выводим на печать комментарий квитирования.

## Enabled

**Синтаксис:**

```
flag = alarm.Enabled
```

**Результат:**

Возвращает **true**, если тревога активна, иначе - **false**.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('alarm')
if not alarm.Enabled and not alarm.Acked then
local err = alarm:Enable('test', 500, 'Enabling...')
if err ~= nil then print(err) end
end
```

В примере получаем тревогу, проверяем её состояние и если она не активна и не квитирована, то активируем ее с указанием источника `'test'`, приоритетом `500` и сообщением `'Enabling...'`. В случае ошибок выводим сообщение об ошибке на печать.

## Severity

**Синтаксис:**

```
number = alarm.Severity
```

**Результат:**

Возвращает приоритет тревоги. Если тревога только создана и не меняла состояние, то приоритет равен нулю.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('alarm')
if alarm.Severity == 0 then
local err = alarm:Enable('test', 500, 'Enabling...')
if err ~= nil then print(err) end
else
local err = alarm:Enable('test', alarm.Severity+100, 'Enabling again...')
if err ~= nil then print(err) end
end
```

В примере получаем тревогу, если приоритет тревоги равен `0`, то активируем её с указанием приоритета `500`, иначе увеличиваем текущий приоритет на `100`. В случае ошибок выводим сообщение об ошибке на печать.

## Message

**Синтаксис:**

```
text = alarm.Message
```

**Результат:**

Возвращает полный текст сообщения тревоги. Если тревога не квитирована, возвращает пустую строку.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
if root ~= nil then
local alarm = root:GetAlarm('Test')
if alarm.Acked then
print(alarm.Message)
end
end
```

В примере получаем тревогу, если она квитирована, выводим на печать полный текст сообщения.

## ReceiveTime

**Синтаксис:**

```
dateTime = alarm.ReceiveTime
```

**Результат:**

Возвращает время возникновения тревоги.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
if root ~= nil then
local alarm = root:GetAlarm('Test')
print(alarm.ReceiveTime)
end
```

В примере получаем тревогу и выводим на печать время ее возникновения.

## SourceName

**Синтаксис:**

```
text = alarm.SourceName
```

**Результат:**

Возвращает имя источника тревоги, заданное методом `Enable()`.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
if root ~= nil then
local alarm = root:GetAlarm('Test')
if alarm.Enabled then
print(alarm.SourceName)
end
end
```

В примере получаем тревогу, если она активна, выводим на печать имя источника тревоги.

# 3.2. Методы

| Метод | Описание |
|---|---|
| *Ack (message)* | Позволяет квитировать аларм и задать сообщение message. |
| *Enable (alarmName, severity, message)* | Позволяет активировать (включить) аларм. |
| *Enable (severity, message)* | Позволяет активировать (включить) аларм без указания источника аларма source. |
| *Disable (severity, message)* | Позволяет отключить аларм. |
| *GetAlarm (alarmName)* | Позволяет получить аларм с заданным именем. |
| *GetEnabledAlarmOrNil (alarmName)* | Позволяет получить активный аларм с заданным именем. Если активного аларма нет, возвращает nil. |
| *GetAlarmsNotAcked ()* | Позволяет получить неквитированные алармы у источника. |
| *GetAlarmsAcked ()* | Позволяет получить квитированные алармы у источника. |
| *GetAlarmsEnabled ()* | Позволяет получить активные (включенные) алармы у источника. |
| *GetAlarmsDisabled ()* | Позволяет получить отключенные алармы у источника. |
| *GetAlarmsCount()* | Позволяет подсчитать общее количество элементов массива алармов. |
| *AckAll (alarmsArray, message)* | Позволяет квитировать все алармы у источника и задать сообщение message. Можно использовать для одного / всех алармов. |
| *DisableAll (alarmsArray, severity, message)* | Позволяет отключить алармы у источника, задать сообщение message и приоритет severity. Можно использовать для одного / всех алармов. |

| **Метод** | **Описание** |
|---|---|
| *EnableAll (alarmsArray, severity, message)* | Позволяет активировать (включить) алармы у источника, задать сообщение message и приоритет severity. Можно использовать для одного / всех алармов. |

## Ack()

**Синтаксис:**

```
err = alarm:Ack(message)
```

**Результат:**

Позволяет квитировать аларм с соответствующим сообщением.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `message` | String | Сообщение, с которым аларм квитируется. |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('instance')
    if alarm.Enabled then
        local err = alarm:Ack('acking...')
            if err ~= nil then
                print(err)
            end
    end
```

В примере получаем аларм с именем `'instance'` в переменную `alarm`.

Если аларм активен (включен) производим квитирование с сообщением - `'acking...'`. В случае ошибки выводим сообщение об ошибке на печать.

## Enable()

**Синтаксис:**

```
err = alarm:Enable(source, severity, message)
```

**Результат:**

Позволяет активировать (включить) аларм. При попытке активации (включения) уже активного (включенного) аларма обновляет его. В случае успеха возвращает *nil*, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `source` | String | Имя источника аларма. |
| `severity` | UInt16 | Приоритет аларма. |
| `message` | String | Сообщение, с которым аларм активируется (включается). |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('alarm')
    if not alarm.Enabled then
        local err = alarm:Enable('test', 500, 'Enabling...')
        if err ~= nil then
            print(err)
        end
    end
```

В примере получаем аларм, проверяем его состояние и если он не активирован (включен), то активируем его с указанием источника `'test'`, приоритетом `500` и сообщением `'Enabling...'`. В случае ошибок выводим сообщение об ошибке на печать.

## Enable()

**Синтаксис:**

```
err = alarm:Enable(severity, message)
```

**Результат:**

Позволяет активировать (включить) аларм без указания источника `source`. При попытке активации (включения) уже активного (включенного) аларма обновляет его. В случае успеха возвращает *nil*, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarm = root:GetAlarm('1')
        alarm:Enable(701, 'TestMethod')
    end
```

В примере в локальную переменную `root` получаем папку **Tags** (источник) алармов, затем в локальную переменную `alarm` получаем аларм и активируем (включаем) его с указанием приоритета - 701 и сообщением - Enabling...

## Disable()

**Синтаксис:**

```
err = alarm:Disable(severity, message)
```

**Результат:**

Отключает активную тревогу. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `severity` | UInt16 | Приоритет события активации тревоги. |
| `message` | String | Сообщение, с которым тревога активируется. |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local alarm = root:GetAlarm('alarm')
    if alarm.Enabled then
        local err = alarm:Disable(100, 'Disabling...')
            if err ~= nil then
                print(err)
            end
    end
```

В примере получаем тревогу, проверяем её состояние и если она активирована, то отключаем с приоритетом - 100 и сообщением - Disabling... В случае ошибок выводим сообщение об ошибке на печать.

## GetAlarm()

**Синтаксис:**

```
alarm = source:GetAlarm(alarmName)
```

**Результат:**

Возвращает объект типа тревогу с указанным именем `alarmName` для источника `source`. Источник тревоги - это серверный объект (агент, привязка, тег, каталог).

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `alarmName` | String | Имя тревоги |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarm = root:GetAlarm('Test')
        alarm:Enable('Test', 500, 'Enabling alarm.')
        alarm:Ack('Acking alarm.')
        print(alarm.Acked)
        print(alarm.AckedBy)
        print(alarm.Enabled)
        print(alarm.Severity)
        print(alarm.Message)
        print(alarm.ReceiveTime)
        print(alarm.Comment)
        print(alarm.SourceName)
        alarm:Disable(300, 'Disabling alarm.')
    end
```

В примере получаем тревогу для источника в переменной `root` (корневой каталог `Tags`), активируем её, квитируем и выводим все свойства на печать. В конце примера отключаем тревогу. Обратите внимание, что ошибки в данном примере не обрабатываются для удобства чтения кода.

## GetEnabledAlarmOrNil()

**Синтаксис:**

```
alarm = source:GetEnabledAlarmOrNil(alarmName)
```

**Результат:**

Возвращает активный аларм `alarm` для источника `source` с именем `alarmName`, если нет активного аларма с именем `alarmName`, возвращает **nil**.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `alarmName` | String | Имя тревоги |

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local alarm = source:GetEnabledAlarmOrNil('Test')
    if alarm ~= nil then
        alarm:Disable(100, 'Disabling...')
    end
```

В примере получаем активный аларм с именем `Test` для корневой папки `Tags`. Если аларм получен, отключаем его.

## GetAlarmsNotAcked()

**Синтаксис:**

```
alarms = source:GetAlarmsNotAcked()
```

**Результат:**

Позволяет получить неаквитированные алармы у источника.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsNotAcked()
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` неквитированные алармы.

## GetAlarmsAcked()

**Синтаксис:**

```
alarms = source:GetAlarmsAcked()
```

**Результат:**

Позволяет получить квитированные алармы у источника.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsAcked()
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` квитированные алармы.

## GetAlarmsEnabled()

**Синтаксис:**

```
alarms = source:GetAlarmsEnabled()
```

**Результат:**

Позволяет получить активные алармы у источника.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsEnabled()
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` активные алармы.

## GetAlarmsDisabled()

**Синтаксис:**

```
alarms = source:GetAlarmsDisabled()
```

**Результат:**

Позволяет получить отключенные алармы у источника.

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsDisabled()
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` отключенные алармы.

## AckAll()

**Синтаксис:**

```
alarm = Context:AckAll(alarmsArray, message)
```

**Результат:**

Позволяет квитировать все алармы у источника и задать сообщение. Можно использовать для одного / всех алармов.

**Пример использования:**

```
-- Квитирование всех активных алармов у источника


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsEnabled()
        Context:AckAll(alarms, "comment")
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` активные (включенные) алармы, которые далее квитируем с сообщением - comment.

## DisableAll()

**Синтаксис:**

```
alarm = Context:DisableAll(alarmsArray, severity, message)
```

**Результат:**

Позволяет отключить алармы у источника, задать сообщение и приоритет. Можно использовать для одного / всех алармов.

**Пример использования:**

```
-- Отключение всех неквитированных алармов у источника


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsNotAcked()
        Context:DisableAll(alarms, 300, "comment")
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` все неквитированные алармы, которые далее отключаем с сообщением - comment и приоритетом - 300.

**Пример использования:**

```
-- Отключение всех квитированных алармов у источника


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsAcked()
        Context:DisableAll(alarms, 300, "comment")
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` все квитированные алармы, которые далее отключаем с сообщением - comment и приоритетом - 300.

## EnableAll()

**Синтаксис:**

```
alarm = Context:EnableAll(alarmsArray, severity, message)
```

**Результат:**

Позволяет активировать алармы у источника, задать сообщение и приоритет. Можно использовать для одного / всех алармов.

**Пример использования:**

```
-- Активация всех отключенных алармов у источника


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarms = root:GetAlarmsDisabled()
        Context:EnableAll(alarms, 700, "comment")
    end
```

В примере получаем в локальную переменную `root` папку **Tags** (источник) с алармами, затем получаем в переменную `alarms` отключенные алармы, которые далее активируем (включаем) с сообщением - comment и приоритетом - 700.

## Дополнительные примеры для работы с алармами

```
-- Вывод ошибки если массив заполнен некорректными данными


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local a = {}
        a[0] = root:GetAlarm('Tessst');
        a[1] = root:GetAlarm('Tessstt');
        a[2] = root:GetAlarm('Tessstttt');
        a[3] = root:GetAlarm('Tesssttttt');
        --наличие в массиве алармов папки - ошибка
        a[4] = Context:GetFolder('/Tags')
        err = Context:AckAll(a, "comment")
            if err ~= nil then
                --вывод сообщения об ошибке
                root:ReportEvent('Event', 300, err)
            end
    end
```

***Прим.:*** *описание метода* ReportEvent ().

```
-- Вывод количества элементов массива алармов


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        --Выбор неквитированных алармов
        local alarms = root:GetAlarmsNotAcked()
        -- подсчет количества алармов
        local cnt = Context:GetAlarmsCount(alarms)
        --вывод в раздел ивентов
        root:ReportEvent('qew', 600, tostring(cnt))
    end

-- Шаблон для генерации алармов с примерами активации (включения) и отключения


local root = Context:GetFolder('/Tags')
    if root ~= nil then
        local alarm = root:GetAlarm('1')
```

```
        alarm:Enable('sos', 701, 'TestMethod')
        local alarm = root:GetAlarm('2')
        alarm:Enable('sos', 702, 'TestMethod')
        local alarm = root:GetAlarm('3')
        alarm:Enable('sos', 703, 'TestMethod')
        local alarm = root:GetAlarm('4')
        alarm:Enable('sos', 704, 'TestMethod')
        alarm:Disable(304, 'TestMethod')
        local alarm = root:GetAlarm('5')
        alarm:Enable('sos', 705, 'TestMethod')
        alarm:Disable(305, 'TestMethod')
        local alarm = root:GetAlarm('6')
        alarm:Enable('sos', 706, 'TestMethod')
        alarm:Disable(306, 'TestMethod')
    end
```

# 4. Управление событиями

Событие представляет собой объект типа **EventItem**. Источником события может быть каталог, тег, агент, привязка (для метода `ReportEvent()`) или пользователь (для метода `ReportUserEvent()`).

## 4.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *AckedWhen* | DateTime | Возвращает отметку времени квитирования события, по умолчанию значение равно `01.01.0001 00:00:00` |
| *AckedWhere* | String | Возвращает место квитирования (второй аргумент метода `Context:AckEvent()`), по умолчанию **nil** |
| *AckedWho* | String | Возвращает имя пользователя, квитировавшего событие, по умолчанию **nil** |
| *ReceiveTime* | DateTime | Возвращает отметку времени получения события |
| *EventId* | Number | Возвращает идентификатор события |
| *SourceId* | Number | Возвращает идентификатор источника события |
| *Time* | DateTime | Возвращает время регистрации события |
| *SourceName* | String | Возвращает имя источника события, заданное методом `ReportEvent()` или `Context:ReportUserEvent()` |
| *Severity* | Number | Возвращает приоритет события |
| *Message* | String | Возвращает текст сообщения события |

## AckedWhen

**Синтаксис:**

```
dateTime = event.AckedWhen
```

**Результат:**

Возвращает отметку времени квитирования события, по умолчанию значение равно `01.01.0001 00:00:00`.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
if event ~= nil then
err = Context:AckEvent(event, 'Event acked from MainRuntime.')
if err ~= nil then error(err) end
print(event.AckedWhen)
end
```

В примере генерируем событие, квитируем и выводим на печать время квитирования.

## AckedWhere

**Синтаксис:**

```
string = event.AckedWhere
```

**Результат:**

Возвращает место квитирования (второй аргумент метода `Context:AckEvent()`). Если событие не квитировано, возвращает **nil**.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
if event ~= nil then
err = Context:AckEvent(event, 'Event acked from MainRuntime.')
if err ~= nil then error(err) end
print(event.AckedWhere)
end
```

В примере генерируем событие, квитируем и выводим на печать сообщение с местом квитирования.

## AckedWho

**Синтаксис:**

```
string = event.AckedWho
```

**Результат:**

Возвращает в виде строки отображаемое и символьное имя пользователя, квитировавшего событие. Если событие не квитировано, возвращает **nil**.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
if event ~= nil then
err = Context:AckEvent(event, 'Event acked from MainRuntime.')
if err ~= nil then error(err) end
print(event.AckedWho)
end
```

В примере генерируем событие, квитируем и выводим на печать отображаемое и символьное имя пользователя, квитировавшего событие.

## ReceiveTime

**Синтаксис:**

```
dateTime = event.ReceiveTime
```

**Результат:**

Возвращает отметку времени получения события.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.ReceiveTime)
```

В примере генерируем событие и выводим на печать время его возникновения.

## EventId

**Синтаксис:**

```
array = event.EventId
```

**Результат:**

Возвращает идентификатор события в виде числа (UInt64).

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.EventId)
```

В примере генерируем событие и выводим на печать длину массива свойства `EventId`.

## SourceId

**Синтаксис:**

```
int = event.SourceId
```

**Результат:**

Возвращает идентификатор источника события в виде числа (UInt32).

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.SourceId)
```

В примере генерируем событие и выводим на идентификатор источника события.

## Time

**Синтаксис:**

```
dateTime = event.Time
```

**Результат:**

Возвращает время регистрации события.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.Time)
```

В примере генерируем событие и выводим на печать время его регистрации.

## SourceName

**Синтаксис:**

```
string = event.SourceName
```

**Результат:**

Возвращает имя источника события, заданное методами `ReportEvent()` или `Context:ReportUserEvent()`.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.SourceName)
```

В примере генерируем событие и выводим на печать имя источника события (в данном примере это `'Event'`).

## Severity

**Синтаксис:**

```
int = event.Severity
```

**Результат:**

Возвращает приоритет события.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.Severity)
```

В примере генерируем событие и выводим на печать приоритет (в нашем примере это значение равно `300`).

## Message

**Синтаксис:**

```
string = event.Message
```

**Результат:**

Возвращает текст сообщения, с которым сгенерировано событие.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
print(event.Message)
```

В примере генерируем событие и выводим на печать текст сообщения (в данном примере это `'Message.'`).

# 4.2. Методы

| Метод | Описание |
|---|---|
| *Context:AckEvent(EventItem, message)* | Квитирует событие |
| *ReportEvent(sourceName, severity, message)* | Генерирует событие и возвращает объект типа **EventItem** |
| *ReportEvent(sourceName, severity, message, ticks)* | Генерирует событие с указанной отметкой времени и возвращает объект типа **EventItem** |
| *Context:ReportUserEvent(sourceName, severity, message)* | Генерирует событие и возвращает объект типа **EventItem**. В качестве источника события используется учетная запись пользователя, запустившего программу |
| *Context:ReportUserEvent(sourceName, severity, message, ticks)* | Генерирует событие с указанной отметкой времени и возвращает объект типа **EventItem**. В качестве источника события используется учетная запись пользователя, запустившего программу |
| *ReadEventHistory(startTicks, endTicks)* | Возвращает коллекцию объектов типа **EventItem** за указанный период для источника, у которого вызван метод |
| *Context:ReadAllEventHistory(startTicks, endTicks)* | Возвращает все события системы в виде коллекции объектов типа **EventItem** за указанный период |

## AckEvent()

**Синтаксис:**

```
err = Context:AckEvent(EventItem, where)
```

**Результат:**

В случае успеха квитирует событие и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `EventItem` | EventItem | Квитируемое событие |
| `where` | String | Сообщение, с которым квитируется событие |

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local event, err = source:ReportEvent('Event', 300, 'Message.')
if event ~= nil then
   err = Context:AckEvent(event, 'Event acked from MainRuntime.')
   if err ~= nil then error(err) end
else
   print(err)
end
```

В примере получаем корневой каталог `Tags`, генерируем событие и если событие сгенерировано без ошибок, то квитируем его.

## ReportEvent()

**Синтаксис:**

```
event, err = source:ReportEvent(sourceName, severity, message)
```

До версии 3.3.19 синтаксис:

```
err, event = source:ReportEvent(sourceName, severity, message)
```

**Результат:**

В случае успеха генерирует событие и возвращает объект типа **EventItem** и **nil**, иначе возвращает **nil** и сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `sourceName` | String | Имя источника события |
| `severity` | Number | Приоритет события |
| `message` | String | Сообщение, с которым генерируется событие |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/Heating/Lines/Line1')
local temperature = folder:GetNode('Temperature')
local hiLimit = folder:GetNode('Hi')
if temperature.Value >= hiLimit.Value then
    local event, err = folder:ReportEvent(folder.DisplayName, 500, 'Температура выше предела')
  if err ~= nil then error('Ошибка генерации события: '..err) end
end
```

В примере получаем папку, получаем два тега из этой папки и сравниваем их значения. Если значение тега с символьным именем `'Temperature` больше или равно значению тега с символьным именем `'Hi'`,то генерируем событие. Объект события для дальнейшей работы не требуется, поэтому обрабатываем только наличие ошибок.

## ReportEvent() с отметкой времени

**Синтаксис:**

```
event, err = source:ReportEvent(sourceName, severity, message, ticks)
```

До версии 3.3.19 синтаксис:

```
err, event = source:ReportEvent(sourceName, severity, message, ticks)
```

**Результат:**

В случае успеха генерирует событие с указанной отметкой времени и возвращает объект типа **EventItem** и **nil**, иначе возвращает **nil** и сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `sourceName` | String | Имя источника события |
| `severity` | Number | Приоритет события |
| `message` | String | Сообщение, с которым генерируется событие |
| `ticks` | Number | Отметка времени возникновения события, в тиках |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/Heating/Lines/Line1')
local tag = folder:GetNode('ContactorOn')
local ticks = 10 ^ 7 -- tikcs per second
local delta = ticks * 60 * 60 * 24 -- ticks per 24 hours
if tag.Value ~= 1 then
   local event, err = folder:ReportEvent(folder.DisplayName, 100, 'Контактор отключен',
 os.time()-delta)
   if err ~= nil then error('Ошибка генерации события: '..err) end
end
```

В примере получаем папку, получаем тег с символьным именем `'ContactorOn'`. Если значение тега не равно `1`, то генерируем событие со смещением на сутки в прошлое. Объект события для дальнейшей работы не требуется, поэтому обрабатываем только наличие ошибок.

## ReportUserEvent()

**Синтаксис:**

```
event, err = Context:ReportUserEvent(sourceName, severity, message)
```

До версии 3.3.19 синтаксис:

```
err, event = Context:ReportUserEvent(sourceName, severity, message)
```

**Результат:**

В случае успеха генерирует событие и возвращает объект типа **EventItem** и **nil**, иначе возвращает **nil** и сообщение об ошибке. В качестве источника события используется учетная запись пользователя, запустившего программу.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `sourceName` | String | Имя источника события |
| `severity` | Number | Приоритет события |
| `message` | String | Сообщение, с которым генерируется событие |

**Пример использования:**

```
local message = InputArguments[0]
if message == nil then return end
local event, err = Context:ReportUserEvent(Context.CurrentUserDisplayName, 400, message)
if err ~= nil then error(err) end
```

В примере запускается программа с одним строковым аргументом, которая генерируем событие. В качестве источника события указываем отображаемое имя пользователя, запустившего программу, полученный аргумент передаем методу в качестве сообщения. Объект события для дальнейшей работы не требуется, поэтому обрабатываем только наличие ошибок.

## ReportUserEvent() с отметкой времени

**Синтаксис:**

```
event, err = Context:ReportUserEvent(sourceName, severity, message, ticks)
```

До версии 3.3.19 синтаксис:

```
err, event = Context:ReportUserEvent(sourceName, severity, message, ticks)
```

**Результат:**

В случае успеха генерирует событие с указанной отметкой времени и возвращает объект типа **EventItem** и **nil**, иначе возвращает **nil** и сообщение об ошибке. В качестве источника события используется учетная запись пользователя, запустившего программу.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `sourceName` | String | Имя источника события |
| `severity` | Number | Приоритет события |
| `message` | String | Сообщение, с которым генерируется событие |
| `ticks` | Number | Отметка времени возникновения события, в тиках |

**Пример использования:**

```
local message = InputArguments[0]
if message == nil then return end
local ticks = 10 ^ 7 -- ticks per second
local date = os.time() - (ticks * 60 * 60 * 24) -- ticks per 24 hours
local event, err = Context:ReportUserEvent(Context.CurrentUserDisplayName, 400, message,
 date)
if err ~= nil then error(err) end
```

В примере запускается программа с одним строковым аргументом, которая генерирует событие со смещением на сутки в прошлое. В качестве источника события указываем отображаемое имя пользователя, запустившего программу, полученный аргумент передаем методу в качестве сообщения. Объект события для дальнейшей работы не требуется, поэтому обрабатываем только наличие ошибок.

## ReadEventHistory()

**Синтаксис:**

```
events, err = source:ReadEventHistory(startTicks, endTicks)
```

**Результат:**

В случае успеха возвращает события в виде коллекции объектов типа **EventItem** в переменную `events` и **nil** в `err` для источника данных `source` (тег, папка, агент, привязка) за период с `startTicks` до `endTicks`. Аргументы `startTicks` и `endTicks` передаются в тиках. В случае ошибки возвращает сообщение об ошибке в `err`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `startTicks` | Number | Начальная отметка времени запроса истории событий, в тиках |
| `endTicks` | Number | Конечная отметка времени запроса истории событий, в тиках |

**Пример использования:**

```
--количество тиков в секунде
local ticks = 10 ^ 7

--интервал в тиках равен 1 час * 60 минут * 60 секунд * тиков в секунду
local interval = 1 * 60 * 60 * ticks

local endx = os.time()
local startx = os.time()-interval

local source = Context:GetFolder('/Tags')
local events, err = source:ReadEventHistory(startx, endx)

if err ~= nil then print(err) return end

local file = io.open('C:/Dev/file.txt', 'w')
for i = 0, events.Length-1 do
   file:write(events[i].Time:ToString()..'\t')
```

```
    file:write(events[i].SourceName..'\t')
    file:write(events[i].Severity..'\t')
    file:write(events[i].Message..'\r\n')
end


file:close()
```

В примере читаем все события источника `source` за последний час и записываем в файл дату и время возникновения события, имя источника, приоритет и сообщение, в конце каждой итерации переходим на новую строку.

## ReadAllEventHistory()

**Синтаксис:**

```
events, err = Context:ReadAllEventHistory(startTicks, endTicks)
```

**Результат:**

В случае успеха возвращает все события в виде коллекции объектов типа **EventItem** в переменную `events` и **nil** в `err` за период с `startTicks` до `endTicks`. Аргументы `startTicks` и `endTicks` передаются в тиках. В случае ошибки возвращает сообщение об ошибке в `err`.

**(!)** Чтение всей истории событий занимает много времени.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `startTicks` | Number | Начальная отметка времени запроса истории событий, в тиках |
| `endTicks` | Number | Конечная отметка времени запроса истории событий, в тиках |

**Пример использования:**

```
--количество тиков в секунде
local ticks = 10 ^ 7

--интервал в тиках равен 1 час * 60 минут * 60 секунд * тиков в секунду
local interval = 1 * 60 * 60 * ticks

local endx = os.time()
local startx = os.time()-interval

local events, err = Context:ReadAllEventHistory(startx, endx)
if err ~= nil then print(err) return end

local file = io.open('C:/Dev/file.txt', 'w')
for i = 0, events.Length-1 do
   file:write(events[i].Time:ToString()..'| ')
   file:write(events[i].SourceName..'| ')
   file:write(events[i].Severity..'| ')
```

```
    file:write(events[i].Message..'\r\n')
end


file:close()
```

В примере читаем все события сервера за последний час и записываем в файл дату и время возникновения события, имя источника, приоритет и сообщение, в конце каждой итерации переходим на новую строку.

# 5. Управление правами / ролями

Управление правом пользователя или роли на изменение каталогов, тегов, агентов и привязок.

**Методы:**

| Метод | Описание |
|---|---|
| *CreateRole(SymbolicName, DisplayName, Description, Password)* | Позволяет создать новую роль в корневом каталоге Roles |
| *Delete()* | Позволяет удалить роль |
| *AddRoleWritePermission(roleSymbolicName)* | Позволяет добавить роли право на изменение объекта |
| *DeleteRoleWritePermission(roleSymbolicName)* | Позволяет удалить у роли право на изменение объекта |
| *AddUserToRole(User,Role)* | Позволяет добавить роль указанному пользователю |
| *DeleteUserFromRole* | Позволяет удалить роль у указанного пользователя |
| *AddRoleExecutePermission ()* | Позволяет добавить роли право на запуск агента (не работает для папок с агентами) |
| *DeleteRoleExecutePermission ()* | Позволяет удалить роль с правом на запуск агента (не работает для папок с агентами) |
| *AddUserExecutePermission ()* | Позволяет добавить пользователю право на запуск агента (не работает для папок с агентами) |
| *DeleteUserExecutePermission ()* | Позволяет удалить пользователя с правом на запуск агента (не работает для папок с агентами) |

## CreateRole()

**Синтаксис:**

```
local role, err = Context:CreateRole(SymbolicName, DisplayName, Description)
```

**Результат:**

Создает роль. В случае успеха возвращает nil, иначе сообщение об ошибке в err.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное имя роли |
| `DisplayName` | String | Отображаемое имя роли |
| `Description` | String | Описание роли |

**Пример использования:**

```
local role, err = Context:CreateRole('TestRoleSN', 'TestRoleDN', 'TestRoleDES')
if err ~= nil then print(err) end
```

В примере создаем новую роль `TestRoleSN` в корневой папке `Roles`.

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

В случае успеха удаляет роль и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Пример использования:**

```
local role1 = Context:GetNode('/Roles/TestRoleSN')
role1:Delete()
```

В примере в локальную переменную `role1` получаем роль `TestRoleSN` из корневого каталога `Roles`, затем удаляем переменную.

## AddRoleWritePermission()

**Синтаксис:**

```
err = source:AddRoleWritePermission(roleSymbolicName)
```

**Результат:**

В случае успеха добавляет роли право на изменение объекта и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `roleSymbolicName` | String | Символьное имя роли, которой добавляется право |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local folders = root:GetFolders()
for i = 0,folders.Length-1 do
   local err = folders[i]:AddRoleWritePermission('engineers')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Tags`, получаем все папки корневого каталога и в цикле добавляем право на изменение для роли с символьным именем `'engineers'`.

## DeleteRoleWritePermission()

**Синтаксис:**

```
err = source:DeleteRoleWritePermission(roleSymbolicName)
```

**Результат:**

В случае успеха удаляет у роли право на изменение объекта и возвращает *nil*, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `roleSymbolicName` | String | Символьное имя роли, у которой удаляется право |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local folders = root:GetFolders()
for i = 0,folders.Length-1 do
   local err = folders[i]:DeleteRoleWritePermission('operators')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Tags`, получаем все папки корневого каталога и в цикле удаляем право на изменение для роли с символьным именем `'operators'`.

## AddUserToRole()

**Синтаксис:**

```
local addUR, err = Context:AddUserToRole(User,Role)
```

**Результат:**

В случае успеха добавляет пользователю роль и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `UserSN` | String | Символьное имя пользователя, которому добавляется роль |
| `RoleSN` | String | Символьное имя роли, добавляемой пользователю |

**Пример использования:**

```
-- пример 1
local user1 = Context:GetNode('/Users/TestSN')
local role1 = Context:GetNode('/Roles/Role')
local addUR, err = Context:AddUserToRole(user1,role1)



-- пример 2
local user1 = Context:GetNode('/Users/TestSN')
local role1 = Context:GetNode('/Roles/r1')
local role1 = Context:GetNode('/Roles/r2')
role1:AddUser(user1)
user1:AddRole(role2)
```

В примере в локальную переменную `user1` из корневой папки `Users` получаем пользователя `TestSN`. Далее в локальную переменную `role1` из корневой папки `Roles` получаем роль `Role` и методом `AddUserToRole()` добавляем пользователю `TestSN` роль `Role`.

## DeleteUserFromRole()

**Синтаксис:**

```
local addUR, err = Context:DeleteUserFromRole(UserSN,RoleSN)
```

**Результат:**

В случае успеха удаляет роль, ранее назначенную пользователю, и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `UserSN` | String | Символьное имя пользователя |
| `RoleSN` | String | Символьное имя роли |

**Пример использования:**

```
local user1 = Context:GetNode('/Users/TestSN')
local role1 = Context:GetNode('/Roles/Role')
local delUR, err = Context:DeleteUserFromRole(user1,role1)
```

В примере в локальную переменную `user1` из корневой папки `Users` получаем пользователя `TestSN`. Далее в локальную переменную `role1` из корневой папки `Roles` получаем роль `Role` и методом `DeleteUserFromRole()` удаляем у пользователя `TestSN` роль `Role`.

## AddRoleExecutePermission()

**Синтаксис:**

```
err = source:AddRoleExecutePermission(RoleSN)
```

**Результат:**

В случае успеха добавляет роли право на запуск агента и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RoleSN` | String | Символьное имя роли |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('test')
   local err = agent:AddRoleExecutePermission('Operators')
   if err ~= nil then print(err)
end
```

В примере в локальную переменную `agent` из корневой папки `Agents` получаем агента `test`. Далее добавляем роли `Operators` право на запуск агента `test`.

## DeleteRoleExecutePermission()

**Синтаксис:**

```
err = source:DeleteRoleExecutePermission(RoleSN)
```

**Результат:**

В случае успеха удаляет у роли право на запуск агента и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RoleSN` | String | Символьное имя роли |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local folders = root:GetNode('test')
   local err = folders:DeleteRoleExecutePermission('Operators')
   if err ~= nil then print(err)
end
```

В примере в локальную переменную `agent` из корневой папки `Agents` получаем агента `test`. Далее удаляем у роли `Operators` право на запуск агента `test`.

## AddUserExecutePermission()

**Синтаксис:**

```
err = source:AddUserExecutePermission(RoleSN)
```

**Результат:**

В случае успеха добавляет пользователю право на запуск агента и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RoleSN` | String | Символьное имя пользователя |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('test')
   local err = agent:AddUserExecutePermission('Operator')
   if err ~= nil then print(err)
end
```

В примере в локальную переменную `agent` из корневой папки `Agents` получаем агента `test`. Далее добавляем пользователю `Operator` право на запуск агента `test`.

## DeleteUserExecutePermission()

**Синтаксис:**

```
err = source:DeleteUserExecutePermission(RoleSN)
```

**Результат:**

В случае успеха удаляет у пользователя право на запуск агента и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RoleSN` | String | Символьное имя пользователя |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('test')
   local err = agent:DeleteUserExecutePermission('Operator')
   if err ~= nil then print(err)
end
```

В примере в локальную переменную `agent` из корневой папки `Agents` получаем агента `test`. Далее удаляем у пользователя `Operator` право на запуск агента `test`.

# 6. Управление пользователями

## 6.1. Методы

| Метод | Описание |
|---|---|
| *CreateUser(SymbolicName, DisplayName, Description, Password)* | Создает нового пользователя в корневом каталоге Users |
| *Delete()* | Удаляет пользователя из корневого каталога Users |
| *AddUserWritePermission(userSymbolicName)* | Добавляет пользователю право на изменение объекта |
| *DeleteUserWritePermission(userSymbolicName)* | Удаляет у пользователя право на изменение объекта |

## CreateUser()

**Синтаксис:**

```
local user, err = Context:CreateUser(SymbolicName, DisplayName, Description, Password)
```

**Результат:**

Создает пользователя. В случае успеха возвращает nil, иначе сообщение об ошибке в err.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное имя пользователя |
| `DisplayName` | String | Отображаемое имя пользователя |
| `Description` | String | Описание пользователя |
| `Password` | String | Пароль пользователя |

**Пример использования:**

```
local user, err = Context:CreateUser('TestSN', 'TestDN', 'TestDES', '')
  if err ~= nil then print(err) end
  print('Done')
```

В примере создаем пользователя с пустым паролем. В случае успеха, печатаем - Done.

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

В случае успеха удаляет пользователя и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Пример использования:**

```
local user1 = Context:GetNode('/Users/TestSN')
user1:Delete()
```

В примере в локальную переменную `user1` получаем пользователя `TestSN` из корневого каталога `Users`, затем удаляем переменную.

## AddUserWritePermission()

**Синтаксис:**

```
err = source:AddUserWritePermission(userSymbolicName)
```

**Результат:**

В случае успеха добавляет пользователю право на изменение объекта и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `userSymbolicName` | String | Символьное имя пользователя, которому добавляется право |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local folders = root:GetFolders()
for i = 0,folders.Length-1 do
local err = folders[i]:AddUserWritePermission('engineer')
if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Tags`, получаем все папки корневого каталога и в цикле добавляем право на изменение для пользователя с символьным именем `'engineer'`.

## DeleteUserWritePermission()

**Синтаксис:**

```
err = source:DeleteUserWritePermission(userSymbolicName)
```

**Результат:**

В случае успеха удаляет у пользователя право на изменение объекта и возвращает **nil**, иначе возвращает сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `userSymbolicName` | String | Символьное имя пользователя, у которого удаляется право |

**Пример использования:**

```
local root = Context:GetFolder('/Tags')
local folders = root:GetFolders()
for i = 0,folders.Length-1 do
local err = folders[i]:DeleteUserWritePermission('operator')
if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Tags`, получаем все папки корневого каталога и в цикле удаляем право на изменение для пользователя с символьным именем `'operator'`.

# 7. Управление каталогами

Каталог представляет собой объект типа `LuaFolder`. API серверных программ позволяет управлять папками в корневых каталогах `Agents` и `Tags`.

## 7.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *SymbolicName* | String | Возвращает символьное имя объекта |
| *DisplayName* | String | Возвращает отображаемое имя объекта |
| *Description* | String | Возвращает описание объекта |
| *Parent* | String | Возвращает родительский объект для тега или папки |

## SymbolicName

**Синтаксис:**

```
string = source.SymbolicName
```

**Результат:**

Возвращает символьное имя объекта, свойство доступно для каталога, тега, агента, привязки. Символьное имя состоит из букв латинского алфавита, цифр и знака `_`, но не может начинаться с цифры. Получение серверных объектов осуществляется по этому свойству.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local folder = source:GetFolder('Heating')
print(folder.SymbolicName)
```

В примере получаем корневой каталог `Tags`, затем получаем из корневого каталога папку с символьным именем `'Heating'` и выводим символьное имя на печать.

## DisplayName

**Синтаксис:**

```
string = source.DisplayName
```

**Результат:**

Возвращает отображаемое имя объекта, свойство доступно для каталога, тега, агента, привязки. Отображаемое имя может содержать любые символы. Для привязки отображаемое имя совпадает с символьным и недоступно для изменения.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local folder = source:GetFolder('Heating')
print(folder.DisplayName)
```

В примере получаем корневой каталог `Tags`, затем получаем из корневого каталога папку с символьным именем `'Heating'` и выводим отображаемое имя на печать.

## Description

**Синтаксис:**

```
string = source.Description
```

**Результат:**

Возвращает описание объекта, свойство доступно для каталога, тега, агента, привязки. Описание может содержать любые символы.

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
local folder = source:GetFolder('Heating')
print(folder.Description)
```

В примере получаем корневой каталог `Tags`, затем получаем из корневого каталога папку с символьным именем `'Heating'` и выводим описание на печать.

## Parent

**Синтаксис:**

```
source.Parent
```

**Результат:**

Возвращает родительский объект для тега или папки `source`. В случае ошибки возвращает **nil**.

**Пример использования метода:**

```
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then print(err) return end
print(tag.Parent.SymbolicName)
print(tag.Parent.DisplayName)
```

В примере получаем тег и выводим на печать символьное и отображаемое имя каталога, в котором находится тег.

# 7.2. Методы

| Метод | Описание |
|---|---|
| *CreateAgentFolder(symbolicName,displayName, description)* | Создает новый каталог в корневом каталоге Agents и вложенных каталогах, возвращает созданный объект |
| *CreateTagFolder(symbolicName,displayName, description)* | Создает новый каталог в корневом каталоге Tags и вложенных каталогах, возвращает созданный объект |
| *CreateFolder(symbolicName, displayName,description)* | Создает новый каталог и возвращает созданный объект |
| *CreateFolders()* | Метод для быстрого создания папок. |
| *MoveFolder()* | Позволяет переместить каталог. |
| *MoveTo()* | |
| *Delete()* | Удаляет каталог |
| *GetFolder(symbolicName)* | Возвращает вложенный каталог |
| *GetFolders()* | Возвращает коллекцию вложенных каталогов |
| *SetSymbolicName(newSymbolicName)* | Устанавливает новое символьное имя |
| *SetDisplayName(newDisplayName)* | Устанавливает новое отображаемое имя |
| *SetDescription(newDescription)* | Устанавливает новое описание |
| *GetNode()* | Возвращает серверный объект |
| *GetNodes()* | Возвращает набор серверных объектов |
| *GetNode(path)* | Возвращает серверный объект |
| *GetFolder(path)* | Возвращает папку |

## CreateAgentFolder()

**Синтаксис:**

```
newFolder, err = folder:CreateAgentFolder(symbolicName, displayName, description)
```

**Результат:**

В случае успеха создает папку в каталоге `folder` и возвращает созданный объект первым аргументом, вторым аргументом возвращает **nil**. В случае ошибки возвращает вторым аргументом сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `symbolicName` | String | Символьное имя нового каталога |
| `displayName` | String | Отображаемое имя нового каталога, может быть пустой строкой |
| `description` | String | Описание нового каталога, может быть пустой строкой |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local folder, err = root:CreateAgentFolder('Project', 'Проект АСУ ТП', '')
if err ~= nil then error(err) end
```

В примере получаем корневой каталог `Agents` и создаем новый каталог с символьным именем `'Project'`, отображаемым именем `'Проект АСУ ТП'` и пустым описанием.

## CreateFolder()

**Синтаксис:**

```
newFolder, err = folder:CreateFolder(SymbolicName, DisplayName, Description)
```

**Результат:**

Создает папку с символьным наименованием `SymbolicName`, отображаемым именем `DisplayName` и описанием `Description` в папке `folder`. В случае успеха возвращает созданную папку в переменную `newFolder` и **nil** в переменную `err`, иначе **nil** в `newFolder` и сообщение об ошибке в `err`. Функция доступна для корневых каталогов `Agents` и `Tags` и вложенных в эти каталоги папки.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное имя нового каталога |
| `DisplayName` | String | Отображаемое имя нового каталога, может быть пустой строкой |
| `Description` | String | Описание нового каталога, может быть пустой строкой |

**Пример использования:**

```
local root = Context:GetFolder('/Tags/Project')
if root ~= nil then
   for i = 1,10,1 do
      local folder, err = root:CreateFolder('Instance'..i, 'Пример'..i, '')
      if err ~= nil then print(err) end
   end
end
```

В примере получаем из корневого каталога `Tags` папку `'Project'` и в полученной папке добавляем `10` нумерованных каталогов с символьным именем `'Instance'` + порядковый номер, отображаемым именем `'Пример'` + порядковый номер и пустым описанием.

## CreateFolders()

**Синтаксис:**

```
CreateFolders('SymbolicName', 1, n)
```

**Результат:**

Создает папки с нумерованным символьным наименованием `SymbolicName`.

**Пример использования:**

```
local g = Context:GetFolder('/Tags/test123/')
g:CreateFolders('folder', 1, 10)
```

В примере в локальную переменную `g` получаем из корневого каталога `Tags` папку `test123` и в полученной папке создаем 10 нумерованных папок с символьным именем `folder` + порядковый номер.

## MoveFolder()

**Синтаксис:**

```
Context:MoveFolder('/Jobs/jobSymbName2','/Jobs/job_folder')
```

**Результат:**

Перемещаем папку `jobSymbName2` в папку `job_folder`.

## MoveTo

**Синтаксис:**

```
local folder = Context:GetFolder('/Jobs/jobSymbName3')

folder:MoveTo('/Jobs/job_folder')
```

**Результат:**

Перемещаем папку `jobSymbName3` в папку `job_folder` через локальную переменную `folder`.

## CreateTagFolder()

**Синтаксис:**

```
newFolder, err = folder:CreateTagFolder(SymbolicName, DisplayName, Description)
```

**Результат:**

Создает папку с символьным наименованием `SymbolicName`, отображаемым именем `DisplayName` и описанием `Description` в папке `folder`. В случае успеха возвращает созданную папку в переменную `newFolder` и **nil** в переменную `err`, иначе **nil** в `newFolder` и сообщение об ошибке в `err`. Функция доступна только для каталога `Tags`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное имя нового каталога |
| `DisplayName` | String | Отображаемое имя нового каталога, может быть пустой строкой |
| `Description` | String | Описание нового каталога, может быть пустой строкой |

**Пример использования:**

```
local root = Context:GetFolder('/Tags/Project')
if root ~= nil then
   for i = 1,10,1 do
      local folder, err = root:CreateTagFolder('Instance'..i, 'Пример'..i, ' ')
      if err ~= nil then print(err) end
   end
end
```

В примере получаем из корневого каталога `Tags` папку `'Project'` и в полученной папке добавляем `10` нумерованных каталогов с символьным именем `'Instance'` + порядковый номер, отображаемым именем `'Пример'` + порядковый номер и пустым описанием.

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

Удаляет `source`, где `source` - папка или тег. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Tags/Project')
   if root ~= nil then
      local tags = root:GetTags()
      for i = 0, tags.Length-1 do
         local err = tags[i]:Delete()
         if err ~= '' then print(err) end
      end
end
```

В примере получаем папку `'Project'` из корневого каталога `Tags`, затем получаем все теги в папке `'Project'` и в цикле удаляем их по одному. Если при удалении возникают ошибки, то выводим их на печать.

## GetFolder()

**Синтаксис:**

```
subfolder, err = folder:GetFolder(SymbolicName)
```

**Результат:**

Возвращает папку в переменную `subfolder` с именем `SymbolicName` из папки `folder` и **nil** в переменную `err`. В случае ошибки возвращает в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное имя папки |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/test')
print(folder.SymbolicName)
```

## GetFolders()

**Синтаксис:**

```
err = folder:GetFolders()
```

**Результат:**

Возвращает коллекцию папок, вложенных в папку `folder`. В случае ошибки возвращает **nil**.

**Пример использования:**

```
local root, err = Context:GetFolder('/Tags')
if err ~= nil then print(err) return end
local folders = root:GetFolders()
if folders == nil then return end
for i = 0, folders.Length-1 do
   if folders[i].SymbolicName == 'Project' then
      print(folders[i].DisplayName)
      return
   end
end
```

В примере получаем все папки корневого каталога `Tags`. В цикле перебираем полученные папки и если символьное имя соответствует условию, то выводим на печать отображаемое имя каталога и выходим из цикла.

## SetSymbolicName()

**Синтаксис:**

```
err = source:SetSymbolicName(newSymbolicName)
```

**Результат:**

Устанавливает символьное наименование `newSymbolicName` серверному объекту в переменной `source`.

В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/Project/OuterValue')
if tag ~= nil then
   local err = tag:SetSymbolicName('InnerValue')
   if err ~= nil then print(err) end
end
```

В примере получаем тег с символьным именем `'OuterValue'` и устанавливаем символьное имя `'InnerValue'`.

## SetDisplayName()

**Синтаксис:**

```
err = source:SetDisplayName(newDisplayName)
```

**Результат:**

Устанавливает отображаемое имя `newDisplayName` серверному объекту `source`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/Project')
if folder ~= nil then
   local err = folder:SetDisplayName('ASDUE Project')
   if err ~= nil then print(err) end
end
```

В примере получаем папку с символьным именем `'Project'` и устанавливаем отображаемое имя `'ASDUE Project'`.

## SetDescription()

**Синтаксис:**

```
err = source:SetDescription(newDescription)
```

**Результат:**

Устанавливает описание `newDescription` серверному объекту `source`. В случае успеха записывает отображаемое имя объекту и возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local folder = Context:GetFolder('/Tags')
local err = folder:SetDescription('Корневая папка с тегами')
if err ~= nil then print(err) end
```

В примере получаем корневой каталог `Tags` и устанавливаем описание `Корневая папка с тегами`.

## GetNode()

**Синтаксис:**

```
node = folder:GetNode(symbolicName)
```

**Результат:**

Возвращает серверный объект с символьным наименованием (symbolicName), указанным в свойстве метода GetNode(). В случае ошибки возвращает **nil**.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `symbolicName` | String | Символьное имя серверного объекта\| |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/test/f1')
local node = folder:GetNode('tag')
print(node.DisplayName)
```

В примере для локальной переменой `folder` получаем папку f1, указав для этого полный путь к ней, в методе GetFolder(). Затем для локальной переменной `node` получаем серверный объект tag, указав символьное имя (symbolicName) объекта в методе GetNode(). Выводим на печать отображаемое имя серверного объекта tag (DisplayName).

## GetNodes()

**Синтаксис:**

```
nodes = folder:GetNodes()
```

**Результат:**

Возвращает набор серверных объектов, вложенных в папку folder. В случае ошибки возвращает **nil**.

**Пример использования:**

```
local root = Context:GetFolder('/Tags/Project')
if root ~= nil then
  local tags = root:GetNodes()
    for i = 0, tags.Length-1 do
    local err = tags[i]:Delete()
      if err ~= nil then print(err) end
   end
end
```

В примере получаем папку Project из корневого каталога Tags, затем получаем все серверные объекты (в данном случае теги) в папке Project и в цикле удаляем их по одному. Если при удалении возникают ошибки, то выводим их на печать.

## GetNode(path)

**Синтаксис:**

```
node = Context:GetNode(nodePath)
```

**Результат:**

Возвращает серверный объект (node), находящийся по пути, указанному в свойстве метода GetNode().

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `nodePath` | String | Путь к серверному объекту |

**Пример использования:**

```
local node = Context:GetNode('/Tags/test/f1/tag')
print(node.DisplayName)
```

В примере для локальной переменной `node` получаем серверный объект tag, для этого указываем полный путь к объекту tag в свойствах метода GetNode(). После выводим на печать отображаемое имя (DisplayName) серверного объекта.

## GetFolder(path)

**Синтаксис:**

```
folder = Context:GetFolder(folderPath)
```

**Результат:**

Возвращает папку (folder), находящуюся по пути, указанному в свойстве метода GetFolder().

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `folderPath` | String | Путь к папке |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/test/')
print(folder.DisplayName)
```

В примере для локальной переменной `folder` получаем папку test, указав в методе GetFolder() полный путь к папке, и выводим на печать отображаемое имя (DisplayName) серверного объекта.

# 8. Управление агентами

## 8.1. Свойства

| Имя | Тип | Описание | MODBUS | OPC DA | OPC UA |
|---|---|---|---|---|---|
| *Active* | Boolean | Возвращает состояние агента | + | + | + |
| *SymbolicName* | String | Возвращает символьное имя агента | + | + | + |
| *DisplayName* | String | Возвращает отображаемое имя агента | + | + | + |
| *Description* | String | Возвращает описание агента | + | + | + |
| *ScanRate* | | Возвращает значение частоты опроса источника данных, в миллисекундах | + | + | + |
| *UseFailAlarm* | Boolean | Возвращает логическое значение состояния генерации аларма при отсутствии связи с источником данных | + | + | + |
| *FailCount* | Number | Возвращает количество сбоев подключения перед генерацией аларма | + | + | + |
| *FailAlarmSeverity* | Number | Возвращает приоритет аларма при сбоях подключения | + | + | + |
| *KeepAliveTimeout* | Number | Возвращает период времени, в течение которого опрашиваемый источник считается доступным, в миллисекундах | - | + | + |
| *Url* | String | Возвращает адрес источника данных | - | + | + |
| *SessionName* | String | Возвращает имя сессии | - | - | + |

| Имя | Тип | Описание | MODBUS | OPC DA | OPC UA |
|---|---|---|---|---|---|
| *User* | String | Возвращает имя пользователя, используемое для подключения к источнику данных | - | - | + |
| *UseSecurity* | Boolean | Возвращает **true**, если используется безопасное подключение к источнику данных, иначе **false** | - | - | + |
| *RequestTimeout* | Number | Возвращает период времени ожидания запроса, в миллисекундах | + | - | - |
| *ConnectTimeout* | Number | Возвращает период ожидания подключения, в миллисекундах | + | - | - |
| *SlaveHost* | String | Возвращает адрес источника данных | + | - | - |
| *SlavePort* | String | Возвращает порт источника данных | + | - | - |
| *SessionTimeout* | Number | Возвращает таймаут сессии (клиент - Сервер OPC UA ) в миллисекундах | - | - | + |

## Active

**Синтаксис:**

```
bool = agent.Active
```

**Результат:**

Возвращает состояние агента `agent`, логическое значение. Если агент активен - **true**, иначе **false**.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   if not agent.Active then
      local err = agent:Start()
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с символьным именем `modbus`, проверяем его состояние и если агент не активен, то запускаем его.

## SymbolicName

**Синтаксис:**

```
string = source.SymbolicName
```

**Результат:**

Возвращает символьное имя объекта, свойство доступно для каталога, тега, агента, привязки. Символьное имя состоит из букв латинского алфавита, цифр и знака "_", но не может начинаться с цифры. Получение серверных объектов осуществляется по этому свойству.

**Пример использования:**

```
local folder = source:GetFolder('/Tags/Heating')
print(folder.SymbolicName)
```

В примере получаем из корневого каталога `Tags` папку с символьным именем `Heating` и выводим символьное имя на печать.

## DisplayName

**Синтаксис:**

```
string = source.DisplayName
```

**Результат:**

Возвращает отображаемое имя объекта, свойство доступно для каталога, тега, агента, привязки. Отображаемое имя может содержать любые символы. Для привязки отображаемое имя совпадает с символьным и недоступно для изменения.

**Пример использования:**

```
local folder = source:GetFolder('/Tags/Heating')
print(folder.DisplayName)
```

В примере получаем из корневого каталога `Tags` папку с символьным именем `Heating` и выводим отображаемое имя на печать.

## Description

**Синтаксис:**

```
string = source.Description
```

**Результат:**

Возвращает описание объекта, свойство доступно для каталога, тега, агента, привязки. Описание может содержать любые символы.

**Пример использования:**

```
local folder = source:GetFolder('/Tags/Heating')
print(folder.Description)
```

В примере получаем из корневого каталога `Tags` папку с символьным именем `Heating` и выводим описание на печать.

## ScanRate

**Синтаксис:**

```
agent.Options.ScanRate
```

**Результат:**

Свойство агента `agent`, которое возвращает значение частоты опроса источника данных, в миллисекундах. Свойство также доступно для агентов OPC DA и OPC UA.

## UseFailAlarm

**Синтаксис:**

```
agent.Options.UseFailAlarm
```

**Результат:**

Свойство агента `agent`. Возвращает логическое значение состояния генерации аларма при отсутствии связи с источником данных.

## FailCount

**Синтаксис:**

```
agent.FailCount
```

**Результат:**

Свойство агента `agent`, которое возвращает количество сбоев подключения перед генерацией аларма.

## FailAlarmSeverity

**Синтаксис:**

```
agent.FailAlarmSeverity
```

**Результат:**

Свойство агента `agent`, которое возвращает приоритет аларма при сбоях подключения.

## KeepAliveTimeout

**Синтаксис:**

```
agent.Options.KeepAliveTimeout
```

**Результат:**

Свойство агента `agent`, которое возвращает период времени, в течение которого опрашиваемый источник считается доступным, в миллисекундах. Свойство доступно для агентов OPC DA и OPC UA.

## Url

**Синтаксис:**

```
agent.Options.Url
```

**Результат:**

Свойство агента `agent`, коотрое возвращает адрес источника данных. Доступно для OPC DA и OPC UA агентов.

## SessionName

**Синтаксис:**

```
agent.Options.SessionName
```

**Результат:**

Свойство OPC UA агента `agent`, которое возвращает имя сессии.

## User

**Синтаксис:**

```
agent.Options.User
```

**Результат:**

Свойство OPC UA агента `agent`, которое возвращает имя пользователя, используемое для подключения к источнику данных.

## UseSecurity

**Синтаксис:**

```
agent.Options.UseSecurity
```

**Результат:**

Свойство OPC UA агента `agent`, которое возвращает **true**, если используется безопасное подключение к источнику данных, иначе **false**.

## RequestTimeout

**Синтаксис:**

```
agent.Options.RequestTimeout
```

**Результат:**

Свойство modbus-агента `agent`. Возвращает период времени ожидания запроса, в миллисекундах.

## ConnectTimeout

**Синтаксис:**

```
agent.ConnectTimeout
```

**Результат:**

Свойство modbus-агента `agent`. Возвращает период ожидания подключения, в миллисекундах.

## SlaveHost

**Синтаксис:**

```
agent.Options.SlaveHost
```

**Результат:**

Свойство modbus-агента `agent`. Возвращает адрес источника данных.

## SlavePort

**Синтаксис:**

```
agent.Options.SlavePort
```

**Результат:**

Свойство modbus-агента `agent`. Возвращает порт источника данных.

## SessionTimeout

**Синтаксис:**

```
agent.Options.SessionTimeout
```

**Результат:**

Возвращает таймаут сессии (клиент - Сервер OPC UA ) в миллисекундах.

**Пример использования:**

```
local agentRoot = Context:GetFolder('/Agents')
local agent = folder:CreateOpcUaAgent('ua_default', 'opc.tcp://192.168.20.215')
-- write
agent.Options:SetSessionTimeout(1234)
-- read
print(agent.Options.SessionTimeout)
```

В примере создаем OPC UA - агент, затем пишем и читаем значение свойства SessionTimeout.

# 8.2. Методы

| Имя | Описание | MODBUS | OPC DA | OPC UA |
|---|---|---|---|---|
| CreateModbus-Agent(symbolicName, host) | Возвращает modbus-агента | + | - | - |
| SetSlaveHost(host) | Устанавливает адрес modbus-агенту | + | - | - |
| SetSymbolicName(new-SymbolicName) | Устанавливает новое символьное имя | + | + | + |
| SetDisplayName(new-DisplayName) | Устанавливает новое отображаемое имя | + | + | + |
| SetDescription(new-Description) | Устанавливает новое описание | + | + | + |
| SetRequest-Timeout(timeout) | Устанавливает период ожидания запроса timeout (в миллисекундах) modbus-агенту | + | - | - |
| SetConnect-Timeout(timeout) | Устанавливает время ожидания подключения timeout (в миллисекундах) modbus-агенту | + | - | - |
| SetScanRate(ScanRate) | Устанавливает значение частоты опроса (в миллисекундах) | + | + | + |
| SetUseFail-Alarm(boolean) | Включает/отключает генерацию тревоги при отсутствии связи с источником данных агента | + | + | + |
| SetFailCount(count) | Устанавливает количество сбоев подключения count агенту | + | + | + |
| SetFailAlarm-Severity(severity) | Устанавливает приоритет тревоги об ошибке подключения severity агента к удаленному источнику данных | + | + | + |
| CreateOpcDa-Agent(symbolicName, host) | Возвращает OPC DA-агента | - | + | - |

| Имя | Описание | MODBUS | OPC DA | OPC UA |
|---|---|---|---|---|
| SetKeepAlive-Timeout(timeout) | Устанавливает период времени агенту, в течение которого опрашиваемый источник считается доступным, в миллисекундах | - | + | + |
| SetUrl(url,useSecurity) | Устанавливает адрес url агенту | - | + | + |
| CreateOpcUa-Agent(SymbolicName, Url) | Возвращает OPC UA-агента | - | - | + |
| SetSession-Name(SessionName) | Устанавливает имя сессии OPC UA-агенту | - | - | + |
| SetUser-Password(user,password) | Устанавливает имя пользователя и пароль для подключения к источнику данных OPC UA | - | - | + |
| Start | Активирует агента | + | + | + |
| Stop | Деактивирует агента | + | + | + |
| Delete() | Удаляет агента или папку с агентами | + | + | + |

## CreateModbusAgent()

**Синтаксис:**

```
modbus, err = folder:CreateModbusAgent('modbusName', 'address_n;address_n+1')
```

**Результат:**

Возвращает modbus-агента `modbus` с именем `modbusName` и адресом `address` / адресами `address_n;address_n+1`, созданного в папке `folder`. В случае успеха возвращает объект агент в переменную `modbus`, в переменную `err` возвращает **nil**. В случае ошибки возвращает в переменную modbus **nil** и в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `modbusName` | String | Символьное имя modbus-агента |
| `address` | String | Адрес modbus-агента |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
   if root ~= nil then
   local agent, err = root:CreateModbusAgent('modbus',
   '192.168.0.100:502;192.168.0.101:503')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents` и в нем создаем агента с именем `modbus` и адресами `192.168.0.100:502` и `192.168.0.101:503`. Созданный агент будет доступен для дальнейшей работы в переменной `agent`, иначе на печать выводим сообщение об ошибке.

***Прим.:*** *Если порт не указан, агент не создается.*

## SetSlaveHost()

**Синтаксис:**

```
err = agent.Options:SetSlaveHost('host')
```

**Результат:**

Устанавливает адрес `host` modbus-агенту `agent`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('modbus')
if agent ~= nil then
   local err = agent.Options:SetSlaveHost('192.168.1.101:501')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents`, агента с символьным именем `modbus` из корневого каталога, устанавливаем новый адрес и указываем порт `192.168.1.101:501`.

***Прим.:*** *Если порт не указан, агент не создается.*

## SetSymbolicName()

**Синтаксис:**

```
err = source:SetSymbolicName(newSymbolicName)
```

**Результат:**

Устанавливает символьное наименование `newSymbolicName` серверному объекту в переменной `source`.

В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
local err = agent:SetSymbolicName('modbus1')
if err ~= nil then print(err) end
end
```

В примере присваиваем агенту с символьным именем `modbus` новое символьное имя `modbus1`.

## SetDisplayName()

**Синтаксис:**

```
err = source:SetDisplayName(newDisplayName)
```

**Результат:**

Устанавливает отображаемое имя `newDisplayName` серверному объекту `source`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
local err = agent:SetDisplayName('modbus1')
if err ~= nil then print(err) end
end
```

В примере присваиваем агенту с символьным именем `modbus` отображаемое имя `modbus1`.

## SetDescription()

**Синтаксис:**

```
err = source:SetDescription(newDescription)
```

**Результат:**

Устанавливает описание `newDescription` серверному объекту `source`. В случае успеха записывает отображаемое имя объекту и возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
local err = agent:SetDescription('modbus agent')
if err ~= nil then print(err) end
```

В примере присваиваем агенту с символьным именем `modbus` описание `modbus agent`.

## SetRequestTimeout()

**Синтаксис:**

```
err = agent.Options:SetRequestTimeout(timeout)
```

**Результат:**

Устанавливает период ожидания запроса `timeout` (в миллисекундах) modbus-агенту `agent`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent, err = root:CreateModbusAgent('modbus','192.168.0.100')
if err ~= nil then print(err) return end
  err = agent.Options:SetRequestTimeout(2000)
    if err ~= nil then print(err)
end
```

В примере создаем агента с символьным именем `modbus` в корневом каталоге `Agents` и устанавливаем период ожидания запроса 2000 мс.

## SetConnectTimeout()

**Синтаксис:**

```
err = agent.Options:SetConnectTimeout(timeout)
```

**Результат:**

Устанавливает время ожидания подключения `timeout` (в миллисекундах) modbus-агенту `agent`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('modbus')
if agent ~= nil then
   local err = agent.Options:SetConnectTimeout(3500)
     if err ~= nil then print(err) end
end
```

В примере получаем `modbus-агента` с символьным именем `modbus` из корневого каталога `Agents` и устанавливаем период ожидания подключения 3500 мс.

## SetScanRate()

**Синтаксис:**

```
err = agent.Options:SetScanRate(time)
```

**Результат:**

Устанавливает значение частоты опроса `time` (в миллисекундах) агенту `agent`. Доступно для всех типов агентов. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `time` | Number | Значение частоты опроса в миллисекундах |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent, err = root:CreateOpcUaAgent('OpcUa','opc.tcp://192.168.0.100:4841')
if agent ~= nil then
   local err = agent.Options:SetScanRate(3000)
      if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents`, создаем агента с символьным именем `OpcUa` и устанавливаем скорость сканирования 3000 мс.

## SetUseFailAlarm()

**Синтаксис:**

```
err = agent.Options:SetUseFailAlarm(bool)
```

**Результат:**

Включает/отключает генерацию тревоги при отсутствии связи с источником данных агента `agent`. Аргумент `bool` - логический. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agents = root:GetNodes()
if agents ~= nil then
   for i = 0, agents.Length-1 do
      local err = agents[i].Options:SetUseFailAlarm(true)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем всех агентов корневого каталога `Agents` и включаем генерацию тревоги при отсутствии связи с источником данных для каждого агента.

## SetFailCount()

**Синтаксис:**

```
err = agent.Options:SetFailCount(count)
```

**Результат:**

Устанавливает количество сбоев подключения count агенту `agent`. При получении `count` сбоев подключения подряд генерируется соответствующий аларм. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agents = root:GetNodes()
if agents ~= nil then
   for i = 0, agents.Length-1 do
      local err = agents[i].Options:SetFailCount(15)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем все агенты корневого каталога `Agents` и устанавливаем количество сбоев подключения перед генерацией аларма равное `15`.

## SetFailAlarmSeverity()

**Синтаксис:**

```
err = agent.Options:SetFailAlarmSeverity(severity)
```

**Результат:**

Устанавливает приоритет тревоги об ошибке подключения `severity` агента `agent` к удаленному источнику данных. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agents = root:GetNodes()
if agents ~= nil then
   for i = 0, agents.Length-1 do
      local err = agents[i].Options:SetFailAlarmSeverity(777)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем все агенты корневого каталога `Agents` и устанавливаем приоритет тревоги ошибки подключения агента к источнику данных `777`.

## CreateOpcDaAgent()

**Синтаксис:**

```
opcda, err = folder:CreateOpcDaAgent('opcdaName','address')
```

**Результат:**

Возвращает OPC DA-агента opcda с именем `opcdaName` и адресом `address`, созданного в папке `folder`. В случае успеха возвращает объект агент в переменную `opcda`, в переменную `err` возвращает **nil**. В случае ошибки возвращает **nil** в переменную `opcda` и в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `opcdaName` | String | Символьное имя OPC DA-агента |
| `address` | String | Адрес OPC DA-агента |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
   if root ~= nil then
    local agent, err = root:CreateOpcDaAgent('OpcDa','opcda://localhost/{32f5756e-...-b140f1160}')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents` и в нем создаем агента с именем `OpcDa` и адресом `opcda://localhost/{32f5756e-...-b140f1160}`. Созданный агент будет доступен для дальнейшей работы в переменной `agent`, иначе на печать выводим сообщение об ошибке.

## SetKeepAliveTimeout()

**Синтаксис:**

```
err = agent.Options:SetKeepAliveTimeout(timeout)
```

**Результат:**

Устанавливает период времени `timeout` агенту `agent`, в течение которого опрашиваемый источник считается доступным, в миллисекундах. Свойство доступно для агентов OPC DA и OPC UA. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent, err = root:CreateOpcUaAgent('OpcDa','opc.tcp://192.168.0.100:4841')
if agent ~= nil then
   local err = agent.Options:SetKeepAliveTimeout(5000)
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents`, создаем агента с символьным именем `OpcDa` и устанавливаем значение параметра `keep alive timeout` 5000 мс.

## SetUrl()

**Синтаксис:**

```
err = agent.Options:SetUrl('url',useSecurity)
```

**Результат:**

Устанавливает адрес `url` агенту `agent`. Доступно для агентов OPC DA и OPC UA. Логический аргумент `useSecurity` заполняется только для агента OPC UA. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `url` | String | Адрес агента |
| `useSecurity` | Boolean | Для OPC DA-агента не заполняется |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetAgent('OpcUa')
if agent ~= nil then
   local err = agent.Options:SetUrl('opc.tcp://192.168.0.100:4841',false)
   if err ~= nil then print(err) end
end
```

В примере получаем OPC UA агента с символьным именем `OpcUa` из корневого каталога `Agents` и устанавливаем адрес `opc.tcp://192.168.0.100:4841` с отключенным защищенным соединением.

## CreateOpcUaAgent()

**Синтаксис:**

```
opcua, err = folder:CreateOpcUaAgent('opcuaName','address')
```

**Результат:**

Возвращает OPC UA-агента `opcua` с именем `opcdaName` и адресом `address`, созданного в папке `folder`. В случае успеха возвращает объект агент в переменную `opcua`, в переменную `err` возвращает **nil**. В случае ошибки возвращает в переменную `opcua` значение **nil** и в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `opcuaName` | String | Символьное имя OPC UA-агента |
| `address` | String | Адрес OPC UA-агента |

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
   if root ~= nil then
   local agent, err = root:CreateOpcUaAgent('OpcUa', 'opc.tcp://192.168.0.100:4840')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents` и в нем создаем агента с именем `OpcUa` и адресом `opc.tcp://192.168.0.100:4840`. Созданный агент будет доступен для дальнейшей работы в переменной `agent`, иначе на печать выводим сообщение об ошибке.

## SetSessionName()

**Синтаксис:**

```
err = agent.Options:SetSessionName('name')
```

**Результат:**

Устанавливает имя сессии `name` OPC UA агенту agent. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local root = Context:GetFolder('/Agents')
local agent, err = root:CreateOpcUaAgent('OpcUa','opc.tcp://192.168.0.100:4841')
if agent ~= nil then
   local err = agent.Options:SetSessionName('KSE Platform')
   if err ~= nil then print(err) end
end
```

В примере получаем корневой каталог `Agents`, создаем агента с символьным именем `OpcUa` и устанавливаем имя сессии `KSE Platform`.

## SetUserPassword()

**Синтаксис:**

```
err = agent.Options:SetUserPassword( 'user ', 'password ')
```

**Результат:**

Устанавливает имя пользователя и пароль для подключения к источнику данных OPC UA агентом `agent`.

Аргументы `user`, `password` - строковые. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования метода:**

```
local root = Context:GetFolder('/Agents')
local agent = root:GetNode('OpcUa')
if agent ~= nil then
   local err = agent.Options:SetUserPassword('operator','')
   if err ~= nil then print(err) end
end
```

В примере получаем OPC UA агента с символьным именем `OpcUa` из корневого каталога `Agents` и устанавливаем имя пользователя `operator` и пустой пароль.

## Start()

**Синтаксис:**

```
err = agent:Start()
```

**Результат:**

Активирует агента `agent`. В случае успеха запускает агента и возвращает **nil**, иначе сообщение об ошибке.

**Пример использования метода:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   if not agent.Active then
   local err = agent:Start()
      if err ~= nil then print(err) end
   end
end
```

## Stop()

**Синтаксис:**

```
err = agent:Stop()
```

**Результат:**

Деактивирует агента `agent`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования метода:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   if agent.Active then
   local err = agent:Stop()
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с символьным именем modbus, проверяем его состояние и если агент активен, то останавливаем его.

## Delete()

**Синтаксис:**

```
source:Delete()
```

**Результат:**

Позволяет удалить папку с агентами, также можно воспользоваться для удаления конкретного агента.

**Пример использования метода:**

```
-- удалим папку с агентом. В примере получаем папку a_test_folder из корневой папки Agents,
затем удаляем папку. В папке создан modbus-агент.
local folder = Context:GetFolder('/Agents/a_test_folder')
folder:Delete()


-- удалим агента. В примере получаем агента test из корневой папки Agents,
затем удаляем агента.
local agent = Context:GetNode('/Agents/test')
agent:Delete()
```

**ВАЖНО!**

Прежде чем удалить агента, следует проверить, что он остановлен (в случае если он не остановлен, оставоить его). Запущенный агент не может быть удален ни в папке, ни отдельно.

**ВАЖНО!**

В случае изменения свойств агента одним из описанных выше методов, можно вывести предупреждение о необходимости перезапуска агента для того, чтобы внесеннные изменения были применены.

Предупреждение будет отображаться в консоли Сервера KSE Platform.

Пример кода см. ниже (меняем адрес modbus-агента):

```
local agent = Context:GetNode('/Agents/ModbusTest')
print("2. Have agent "..agent.DisplayName)
if agent ~= nil then
 local err, warn = agent.Options:SetSlaveHost('127.0.1.55:111')
 if err ~= nil then print(err) end
```

```
 if warn ~= nil then print(warn) end
end
```

# 9. Управление привязками

## 9.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *Address* | String | Возвращает адрес привязки |
| *ClampHigh* | Boolean | Возвращает **true** , если у привязки установлен соответствующий флаг в настройках масштабирования |
| *ClampLow* | Boolean | Возвращает **true** , если у привязки установлен соответствующий флаг в настройках масштабирования |
| *DisplayName* | String | Возвращает отображаемое имя объекта ( DisplayName совпадает с SymbolicName и недоступно для изменения) |
| *Description* | String | Возвращает описание объекта |
| *SymbolicName* | String | Символьное наименование привязки |
| *Gain* | Number | Возвращает коэффициент масштабирования |
| *Offset* | Number | Возвращает коэффициент смещения читаемого значения |
| *RawHigh* | Number | Возвращает верхнее допустимое значение читаемого регистра или тега |
| *RawLow* | Number | Возвращает нижнее допустимое значение читаемого регистра или тега |
| *Read* | Boolean | Возвращает **true**, если производится чтение регистра или тега, иначе **false** |
| *ScaledHigh* | Number | Возвращает коэффициент масштабирования верхнего допустимого значения читаемого регистра или тега |
| *ScaledLow* | Number | Возвращает коэффициент масштабирования нижнего допустимого значения читаемого регистра или тега |

| Имя | Тип | Описание |
|---|---|---|
| *ScalingMode* | String | Возвращает режим масштабирования читаемого регистра или тега |
| *TagPath* | String | Возвращает путь к серверному тегу |
| *Write* | Boolean | Возвращает **true**, если доступна запись значения в регистр или тег источника данных, иначе **false** |
| *WriteNullOnFail* | Boolean | Возвращает **true**, если при отсутствии связи с источником данных в серверный тег записывается значение **nil** со статусом Uncertain , иначе **false** |

## Address

**Синтаксис:**

```
binding.Address
```

**Результат:**

Возвращает адрес привязки.

## ClampHigh

**Синтаксис:**

```
binding.ClampHigh
```

**Результат:**

Возвращает **true**, если у привязки установлен соответствующий флаг в настройках масштабирования. Иначе возвращает **false**.

## ClampLow

**Синтаксис:**

```
binding.ClampLow
```

**Результат:**

Возвращает **true**, если у привязки установлен соответствующий флаг в настройках масштабирования. Иначе возвращает **false**.

## DisplayName

**Синтаксис:**

```
string = source.DisplayName
```

**Результат:**

Возвращает отображаемое имя объекта, свойство доступно для каталога, тега, агента, привязки. Отображаемое имя может содержать любые символы. Для привязки отображаемое имя совпадает с символьным и недоступно для изменения.

**Пример использования:** описан ранее см. *DisplayName*

## Description

**Синтаксис:**

```
string = source.Description
```

**Результат:**

Возвращает описание объекта, свойство доступно для каталога, тега, агента, привязки. Описание может содержать любые символы.

**Пример использования:** описан ранее см. *Description*

## SymbolicName

**Синтаксис:**

```
string = source.SymbolicName
```

**Результат:**

Возвращает символьное имя объекта, свойство доступно для каталога, тега, агента, привязки. Символьное имя состоит из букв латинского алфавита, цифр и знака "_", но не может начинаться с цифры. Получение серверных объектов осуществляется по этому свойству.

**Пример использования:** описан ранее см. *SymbolicName*

## Gain

**Синтаксис:**

```
binding.Gain
```

**Результат:**

Возвращает коэффициент масштабирования.

## Offset

**Синтаксис:**

```
binding.Offset
```

**Результат:**

Возвращает коэффициент смещения читаемого значения.

## RawHigh

**Синтаксис:**

```
binding.RawHigh
```

**Результат:**

Возвращает верхнее допустимое значение читаемого регистра или тега. Применяется при линейном и квадратичном масштабировании.

## RawLow

**Синтаксис:**

```
binding.RawLow
```

**Результат:**

Возвращает нижнее допустимое значение читаемого регистра или тега. Применяется при линейном и квадратичном масштабировании.

## Read

**Синтаксис:**

```
binding.Read
```

**Результат:**

Возвращает **true**, если производится чтение регистра или тега, иначе **false**.

## ScaledHigh

**Синтаксис:**

```
binding.ScaledHigh
```

**Результат:**

Возвращает коэффициент масштабирования верхнего допустимого значения читаемого регистра или тега. Применяется при линейном и квадратичном масштабировании.

## ScaledLow

**Синтаксис:**

```
binding.ScaledLow
```

**Результат:**

Возвращает коэффициент масштабирования нижнего допустимого значения читаемого регистра или тега. Применяется при линейном и квадратичном масштабировании.

## ScalingMode

**Синтаксис:**

```
binding.ScalingMode
```

**Результат:**

Возвращает режим масштабирования читаемого регистра или тега. По умолчанию масштабирование отсутствует. Возможны следующие значения:

- масштабирование отсутствует, см. *SetNoScaling*;
- линейное, см. *SetLinearScaling*;
- квадратичное, см. *SetSquareRootScaling*;
- коэффициент и смещение, см. *SetGainOffsetScaling*.

## TagPath

**Синтаксис:**

```
binding.TagPath
```

**Результат:**

Возвращает путь к серверному тегу.

## Write

**Синтаксис:**

```
binding.Write
```

**Результат:**

Возвращает **true**, если доступна запись значения в регистр или тег источника данных, иначе **false**.

## WriteNullOnFail

**Синтаксис:**

```
binding.WriteNullOnFail
```

**Результат:**

Возвращает **true**, если при отсутствии связи с источником данных в серверный тег записывается значение **nil**со статусом `Uncertain`, иначе **false**.

# 9.2. Методы

| Имя | Описание |
|---|---|
| *CreateBinding(Symbolic-Name,Adress,Read,Write,TagPath)* | Возвращает привязку агента |
| *GetBinding(SymbolicName)* | Возвращает привязку с символьным наименованием агента |
| *GetBindings()* | Возвращает коллекцию привязок агента |
| *SetActive(bool)* | Активирует/деактивирует привязку |
| *SetAddress(Address)* | Устанавливает адрес единицы источника данных (регистра или тега) привязке |
| *SetGainOffsetScaling(Gain, Offset)* | Устанавливает масштабирование привязке |
| *SetDescription(newDescription)* | Устанавливает новое описание |
| *SetLinearScaling(RawHigh, RawLow, ScaledHigh, ScaledLow, ClampHigh, ClampLow)* | Устанавливает линейное масштабирование привязке |
| *SetNoScaling()* | Отключает масштабирование привязки |
| *SetReadWrite(Read, Write)* | Устанавливает привязке доступ на чтение и запись в регистр или тег |
| *SetSquareRootScaling(RawHigh, RawLow, ScaledHigh, ScaledLow, ClampHigh, ClampLow)* | Устанавливает квадратичное масштабирование привязке |
| *SetTagPath(TagPath)* | Устанавливает путь к серверному тегу привязке |
| *SetWriteNullOnFail(bool)* | Устанавливает привязке логический параметр на запись в серверные теги |

## CreateBinding()

**Синтаксис:**

```
binding, err = agent:CreateBinding(bindingName,address,read,write,tagPath)
```

**Результат:**

Возвращает привязку `binding` агента `agent`, где:

- `bindingName` - имя привязки,
- `address` - адрес,
- `read` и `write` - доступ на чтение и запись, логические значения,
- `tagPath` - путь к серверному тегу.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное наименование привязки |
| `Adress` | String | Адрес привязки |
| `Read` | Boolean | **true**, если производится чтение регистра или тега, иначе **false** |
| `Write` | Boolean | **true**, если доступна запись значения в регистр или тег источника данных, иначе **false** |
| `TagPath` | String | Путь к серверному тегу |

**Пример использования (для Modbus-агента):**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding, err = agent:CreateBinding(test,1:4:0:1::,true,false,/Tags/tag1)
   if err ~= nil then print(err) end
end
```

В примере получаем агента с именем `modbus` и создаем привязку с именем `test`, адресом `1:4:0:1::`, доступом только на чтение значения из регистра в тег `tag1`.

**Пример использования (для OpcUA-агента):**

```
local agent = Context:GetNode('/Agents/OpcUa')
agent:CreateBinding('tagA_1','ns=2;s=KSE_OPCUA_TEST.KSEopcUA.tag_1',true,false,
'/Tags/Project/KEP_10k/tagA_1')
```

В примере получаем агента с именем `OpcUa` и создаем привязку с именем `tagA_1`, адресом `ns=2;s=KSE_OPCUA_TEST.KSEopcUA.tag_1`, доступом только на чтение значения в тег `/Tags/Project/KEP_10k/tagA_1`.

## GetBinding()

**Синтаксис:**

```
binding = agent:GetBinding(SymbolicName)
```

**Результат:**

Возвращает привязку с символьным наименованием `SymbolicName` агента `agent`. В случае ошибки возвращает **nil**. Кроме свойств `SymbolicName`, `DisplayName`, `Description` у объекта привязка доступны следующие свойства:

- `Active`: если привязка активна (осуществляется получение и/или запись), возвращает **true**, иначе **false**;
- `Address`: адрес единицы данных удаленного источника в виде строки;
- `Read`: если доступ на чтение включен, возвращает **true**, иначе **false**;
- `Write`: если доступ на запись включен, возвращает **true**, иначе **false**;
- `ScalingMode`: режим масштабирования значения:
  - 0 - без масштабирования;
  - 1 - линейное;
  - 2 - квадратичное;
  - 3 - коэффициент и смещение.
- `RawHigh`: верхняя допустимая граница значения в удаленном источнике;
- `RawLow`: нижняя допустимая граница значения в удаленном источнике;
- `ScaledHigh`: масштабируемая верхняя граница значения;
- `ScaledLow`: масштабируемая нижняя граница значения;
- `ClampHigh`: усечение значения к верхней границе, если значение выше допустимого предела;
- `ClampLow`: усечение значения к нижней границе, если значение ниже допустимого предела;
- `Gain`: коэффициент масштабирования;
- `Offset`: коэффициент смещения читаемого;
- `TagPath`: путь к тегу, с которым связана привязка;
- `WriteNullOnFail`: запись значения **null** при отсутствии связи с удаленным источником или при остановке агента.

**Внимание:**метод `SetDisplayName` не поддерживается для данного типа объектов и при попытке вызова вернет ошибку `BadNotSupported`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Символьное наименование привязки |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      print(binding.SymbolicName)
      print(binding.DisplayName)
      print(binding.Description)
      print(binding.Active)
      print(binding.Address)
      print(binding.Read)
      print(binding.Write)
      print(binding.ScalingMode)
      print(binding.RawHigh)
      print(binding.RawLow)
      print(binding.ScaledHigh)
      print(binding.ScaledLow)
      print(binding.ClampHigh)
      print(binding.ClampLow)
      print(binding.Gain)
      print(binding.Offset)
      print(binding.TagPath)
      print(binding.WriteNullOnFail)
   end
end
```

В примере получаем агента, привязку и выводим все доступные свойства привязки на печать.

## GetBindings()

**Синтаксис:**

```
err = agent:GetBindings()
```

**Результат:**

Возвращает коллекцию привязок агента `agent`. В случае ошибки возвращает **nil**.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local bindings = agent:GetBindings()
   if bindings == nil then return end
   for i = 0, bindings.Length-1 do
      local err = bindings[`i`]:SetActive(false)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем все привязки агента с символьным именем `modbus` и в цикле отключаем активность каждой привязки (получение единицы данных с удаленного источника).

## SetActive()

**Синтаксис:**

```
err = binding:SetActive(bool)
```

**Результат:**

Активирует/деактивирует привязку `binding`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local bindings = agent:GetBindings()
   if bindings ~= nil then
      for i = 0, bindings.Length-1 do
         local err = bindings[`i`]:SetActive(false)
         if err ~= nil then print(err) end
      end
   end
end
```

В примере получаем агента с именем `modbus`, затем получаем все привязки этого агента и в цикле отключаем у каждой привязки активность.

## SetAddress()

**Синтаксис:**

```
err = binding:SetAddress(address)
```

**Результат:**

Устанавливает адрес единицы источника данных (регистра или тега) привязке `binding`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `Address` | String | Адрес привязки |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local bindings = agent:GetBindings()
   if bindings ~= nil then
      for i = 0, bindings.Length-1 do
         local err = bindings[`i`]:SetAddress(1:4:..i..:1::)
         if err ~= nil then print(err) end
      end
   end
end
```

В примере получаем агента с именем `modbus`, затем получаем все привязки этого агента и в цикле устанавливаем адрес единицы данных удаленного источника для каждой привязки (чтение слова из таблицы `Holding Registers` с нулевого регистра из modbus-устройства с адресом 1).

## SetGainOffsetScaling()

> **ВАЖНО!**
>
> Подробнее о вычислениях при масштабировании см. в документе **Руководство по Среде разработки Studio → Глава 13. Приложение №1 Масштабирование привязок**.

**Синтаксис:**

```
err = binding:SetGainOffsetScaling(Gain, Offset)
```

**Результат:**

Устанавливает масштабирование привязке `binding` с помощью коэффициента `gain` и смещения `offset`.

В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `Gain` | Number | Коэффициент масштабирования |
| `Offset` | Number | Коэффициент смещения читаемого значения |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetGainOffsetScaling(2, 3)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем коэффициент масштабирования `x2` и смещение на +3.

## SetDescription()

**Синтаксис:**

```
err = source:SetDescription(newDescription)
```

**Результат:**

Устанавливает описание `newDescription` серверному объекту `source`. В случае успеха записывает отображаемое имя объекту и возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** описан ранее см. *SetDescription()*

## SetLinearScaling()

**Синтаксис:**

```
err = binding: SetLinearScaling(rawHigh,rawLow,scaledHigh,scaledLow,clampHigh,clampLow)
```

**Результат:**

Устанавливает линейное масштабирование привязке `binding`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RawHigh` | Number | Верхнее допустимое значение читаемого регистра или тега |
| `RawLow` | Number | Нижнее допустимое значение читаемого регистра или тега |
| `ScaledHigh` | Number | Коэффициент масштабирования верхнего допустимого значения читаемого регистра или тега |
| `ScaledLow` | Number | Коэффициент масштабирования нижнего допустимого значения читаемого регистра или тега |
| `ClampHigh` | Boolean | **true**, если у привязки установлен соответствующий флаг в настройках масштабирования |
| `ClampLow` | Boolean | **true**, если у привязки установлен соответствующий флаг в настройках масштабирования |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetLinearScaling(8000,-2000,800,-200,true,true)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем линейное масштабирование с верхней границей читаемого значения `8000`, нижней границей читаемого значения `-2000`, верхней границей масштабируемого значения `800`, нижней границей масштабируемого значения `-200`, с усечением значений при выходе за пределы диапазона.

## SetNoScaling()

**Синтаксис:**

```
err = binding:SetNoScaling()
```

**Результат:**

Отключает масштабирование привязки `binding`. При этом все параметры масштабирования сохраняются и применяются при возобновлении масштабирования. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetNoScaling()
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и отключаем масштабирование.

## SetSquareRootScaling()

**Синтаксис:**

```
err = binding:SetSquareRootScaling(rawHigh,rawLow,scaledHigh,scaledLow,clampHigh,clampLow)
```

**Результат:**

Устанавливает квадратичное масштабирование привязке `binding`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `RawHigh` | Number | Верхнее допустимое значение читаемого регистра или тега |
| `RawLow` | Number | Нижнее допустимое значение читаемого регистра или тега |
| `ScaledHigh` | Number | Коэффициент масштабирования верхнего допустимого значения читаемого регистра или тега |
| `ScaledLow` | Number | Коэффициент масштабирования нижнего допустимого значения читаемого регистра или тега |
| `ClampHigh` | Boolean | **true**, если у привязки установлен соответствующий флаг в настройках масштабирования |
| `ClampLow` | Boolean | **true**, если у привязки установлен соответствующий флаг в настройках масштабирования |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
local binding = agent:GetBinding(binding1)
if binding ~= nil then
local err = binding:SetSquareRootScaling(10000,0,100,0,false,false)
if err ~= nil then print(err) end
end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем квадратичное масштабирование с верхней границей читаемого значения `10000`, нижней границей читаемого значения `0`, верхней границей масштабируемого значения `100`, нижней границей масштабируемого значения `0`, без усечения значений при выходе за пределы диапазона.

## SetReadWrite()

**Синтаксис:**

```
err = binding:SetReadWrite(Read, Write)
```

**Результат:**

Устанавливает привязке `binding` доступ на чтение и запись в регистр или тег. Аргументы `read` и `write` - логические. В случае успеха возвращает nil, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `Read` | Boolean | **true**, если производится чтение регистра или тега, иначе **false** |
| `Write` | Boolean | **true**, если доступна запись значения в регистр или тег источника данных, иначе **false** |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetReadWrite(true, false)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем доступ к единице данных удаленного источника только на чтение.

## SetTagPath()

**Синтаксис:**

```
err = binding:SetTagPath(tagPath)
```

**Результат:**

Устанавливает путь к серверному тегу `tagPath` привязке `binding`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `TagPath` | String | Путь к тегу |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetTagPath(/Tags/Project/tag1)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем путь к тегу `/Tags/Project/tag1`.

## SetWriteNullOnFail()

**Синтаксис:**

```
err = binding:SetWriteNullOnFail(bool)
```

**Результат:**

Устанавливает привязке `binding` логический параметр на запись в серверные теги значения **nil** со статусом `Uncertain` при отсутствии связи с источником данных. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `bool` | Boolean | **true** - включить, **false** - отключить |

**Пример использования:**

```
local agent = Context:GetNode('/Agents/modbus')
if agent ~= nil then
   local binding = agent:GetBinding(binding1)
   if binding ~= nil then
      local err = binding:SetWriteNullOnFail(true)
      if err ~= nil then print(err) end
   end
end
```

В примере получаем агента с именем `modbus`, затем у этого агента получаем привязку с именем `binding1` и устанавливаем **true** для записи в значение тега **null** при ошибках связи и остановке агента.

# 10. Управление тегами

## 10.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *Value* | Object | Возвращает значение тега |
| *StatusCode* | Number | Возвращает код статуса значения |
| *Timestamp* | DateTime | Возвращает отметку времени получения значения тега |
| *SymbolicName* | String | Возвращает символьное имя объекта |
| *DisplayName* | String | Возвращает отображаемое имя объекта |
| *Description* | String | Возвращает описание объекта |
| *Expression* | String | Возвращает выражение тега, если тег вычисляемый. Иначе возвращает пустую строку |
| *TargetPath* | String | Возвращает путь к целевому тегу, если тег ссылается на другой тег. Иначе возвращает пустую строку |
| *BindingPath* | String | Возвращает путь к привязке, если на тег ссылается привязка. Иначе возвращает пустую строку |
| *Parent* | LuaFolder | Возвращает объект - родительскую папку тега |

## Value

**Синтаксис:**

```
value = tag.Value
```

**Результат:**

Возвращает значение тега.

**Пример использования:**

```
local path = InputArguments[0]
local tag1 = Context:GetNode(path)
if tag1 ~= nil then
   local folder = tag1.Parent
   if folder ~= nil then
      local tag2 = folder:GetNode('tag2')
      if tag2 ~= nil then
         local err = tag2:SetValue(tag1.Value)
         if err ~= ' ' then print(err) end
      end
   end
end
```

Пример представлен для программы, которая запускается по изменению тега, в качестве нулевого аргумента передается путь к тегу, инициировавшему запуск программы. С помощью нулевого аргумента получаем объект тег в переменную `tag1`, с помощью свойства `Parent` получаем объект папку в переменную `folder`, в котором находится `tag1`. Из каталога получаем второй тег в переменную `tag2` и записываем значение первого тега.

## StatusCode

**Синтаксис:**

```
status = tag.StatusCode
```

**Результат:**

Возвращает код статуса значения.

**Пример использования:**

```
local t = Context:GetNode('/Tags/APILua/tag1')
if t.StatusCode ~= 0 then
t:ReportEvent(t.SymbolicName, 700, 'Bad status code: '..tostring(t.StatusCode))
else
t:ReportEvent(t.SymbolicName, 100, 'Good status code')
end
```

В примере получаем тег, проверяем код статуса значения и генерируем соответствующее событие (значение `0` соответствует статусу `Good`)

## Timestamp

**Синтаксис:**

```
dateTime = tag.Timestamp
```

**Результат:**

Возвращает отметку времени получения значения тега.

**Пример использования:**

```
local t = Context:GetNode('/Tags/APILua/tag1')
t:ReportEvent(t.SymbolicName,               100,               'Value               was
 received:'..t.Timestamp:ToString('yyyy.MM.dd HH:mm:ss'))
```

В примере получаем тег и генерируем событие. В сообщении события выводим отметку времени в указанном формате. Метод `ToString()` доступен для объектов типа userdata, которым является значение свойства. Вызов метода `ToString()` возможен и без указания аргументов.

## SymbolicName

**Синтаксис:**

```
text = tag.SymbolicName
```

**Результат:**

Возвращает символьное имя тега. Символьное имя состоит из букв латинского алфавита, цифр и знаков _, но не может начинаться с цифры. Тег однозначно идентифицируется в папке по этому свойству. Пример использования смотрите в описании свойства *Timestamp*.

## DisplayName

**Синтаксис:**

```
text = tag.DisplayName
```

**Результат:**

Возвращает отображаемое имя тега. Отображаемое имя может содержать любые символы.

**Пример использования:**

```
local t = Context:GetNode('/Tags/APILua/tag1')

t:ReportEvent(t.SymbolicName, 100, t.DisplayName)
```

В примере получаем тег и генерируем событие, в сообщении выводим отображаемое имя тега.

## Description

**Синтаксис:**

```
text = tag.Description
```

**Результат:**

Возвращает описание тега. Может содержать любые символы.

**Пример использования:**

```
local t = Context:GetNode('/Tags/APILua/tag1')
if t.Description == nil or t.Description == '' then return end
t:ReportEvent(t.SymbolicName, 100, t.Description)
```

В примере получаем тег и если его описание не равно **nil** или пустой строке, то генерируем событие, в сообщении выводим описание тега.

## Expression

**Синтаксис:**

```
text = tag.Expression
```

**Результат:**

Возвращает выражение тега, если тег вычисляемый. Иначе возвращает пустую строку.

**Пример использования:**

```
local expression = 'return #[/Tags/APILua/tag2]#.ValueOrDefault'
local t = Context:GetNode('/Tags/APILua/tag1')
if t.Expression == '' then
t:SetExpression(expression)
end
```

В примере получаем тег, если у тега отсутствует выражение, то устанавливаем новое выражение.

## TargetPath

**Синтаксис:**

```
text = tag.TargetPath
```

**Результат:**

Возвращает путь к целевому тегу, если тег ссылается на другой тег. Иначе возвращает пустую строку.

**Пример использования:**

```
local path = '/Tags/APILua/tag1'
local t = Context:GetNode('/Tags/APILua/tag3')
if t.TargetPath == '' then
 local err = t:SetTargetPath(path)
 if err ~= nil then error(err) end
end
```

В примере получаем тег, если у тега не задано значение свойства `TargetPath`, то устанавливаем новое значение.

## BindingPath

**Синтаксис:**

```
text = tag.BindingPath
```

**Результат:**

Возвращает путь к привязке, если на тег ссылается привязка. Иначе возвращает пустую строку.

**Пример использования:**

```
local t = Context:GetNode('/Tags/APILua/tag1')
if t.BindingPath ~= '' then
print(t.SymbolicName..': '..t.BindingPath)
else
print(t.SymbolicName..': BindingPath is empty.')
end
```

В примере получаем тег и проверяем значение свойства `BindingPath`. В зависимости от значения свойства выводим на печать соответствующее сообщение.

## Parent

**Синтаксис:**

```
local parentFolder = tag.Parent
```

**Результат:**

Возвращает объект - родительскую папку тега.

# 10.2. Методы

| Имя | Описание |
|---|---|
| *GetAggregateTag(symbolicName)* | Возвращает тег-агрегат с заданным символьным наименованием |
| *Reaggregate()* | Позволяет пересчитать значения агрегатов за определенный период времени |
| *SetExpression(expression)* | Устанавливает выражение тегу, тег становится вычисляемым |
| *SetTargetPath(path)* | Устанавливает ссылку на тег по заданному пути |
| *SetTargetTag(tag)* | Устанавливает ссылку на заданный тег |
| *CreateTag(symbolicName, displayName, description, templatePath)* | Создает тег |
| *CreateTags()* | Метод для быстрого создания тегов |
| *SetSymbolicName(newName)* | Устанавливает символьное наименование тегу |
| *SetDisplayName(newName)* | Устанавливает отображаемое имя тегу |
| *SetDescription(newDescription)* | Устанавливает описание тегу |
| *SetValue(value)* | Устанавливает тегу заданное значение с нулевым кодом состояния (Good) |
| *SetValue(value, statusCode)* | Устанавливает значение с заданным статусом тегу |
| *SetValue(value, statusCode, timestamp)* | Устанавливает тегу значение с заданным кодом состояния и отметкой времени |
| *ReadHistory(startTicks, endTicks, bounds)* | Возвращает историю значений тега за определенный период и пустую строку в err |
| *ReadProcessedHistory(startTicks, endTicks, function, interval, uncertainAsBad, percentBad,* | Возвращает агрегированные значения тега за определенный период с интервалом агрегирования |

| Имя | Описание |
| --- | --- |
| percentGood, slopedExtrapolation, steppedInterpolation) | |

## GetAggregateTag()

**Синтаксис:**

```
aggregate = tag:GetAggregateTag(symbolicName)
```

**Результат:**

Возвращает тег-агрегат с символьным наименованием тега. Если тег-агрегат не найден, возвращает **nil**.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/APILua/tag1')
local aggregate = tag:GetAggregateTag('AverageDaily')
if aggregate ~= nil then
   print('value:'..tostring(aggregate.Value))
   print('date&time:'..aggregate.Timestamp:ToString())
else
   print('aggregate not found')
end
```

В примере получаем суточный тег-агрегат `AverageDaily` тега с символьным именем `tag1` из корневого каталога `Tags` и выводим на печать текущее значение и отметку времени сохранения значения агрегата.

## Reaggregate()

**Синтаксис:**

```
local err = aggregate:Reaggregate(begin, end)
```

**Результат:**

Позволяет пересчитать значения агрегатов за определенный период времени.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/Project/HalfHourly/int_2')
print(tag.Value)
local                                    aggregateTags                                    =
 {"AverageHalfHourly", "CountHalfHourly", "MinimumHalfHourly", "TotalHalfHourly"}
for i = 1, #aggregateTags , 1 do
local aggregate = tag:GetAggregateTag( aggregateTags[i])
local beginPoint = '11-12-2019 11:00:00'
local endPoint =  '11-12-2019 15:30:00'
local err = aggregate:Reaggregate(beginPoint, endPoint)
  if err ~= nil then
    print('reaggregate error')
  else
    print('reaggregate success')
  end
end
```

В примере за указанный период `beginPoint` и `endPoint` пересчитываются значения агрегатов `AverageHalfHourly`, `CountHalfHourly`, `MinimumHalfHourly`, `TotalHalfHourly`.

## SetExpression()

**Синтаксис:**

```
err = tag2:SetExpression(expression)
```

**Результат:**

Устанавливает выражение для вычисляемого тега. Возвращает **nil** в случае успеха или текст ошибки при неудаче.

**Пример использования:**

```
local expression = 'return #[/Tags/tag1]#.Value'
local tag2 = Context:GetNode('/Tags/tag2')
local err = tag2:SetExpression(expression)
if err ~= nil then error(err) end
```

## SetTargetPath()

**Синтаксис:**

```
err = tag:SetTargetPath(tagPath)
```

**Результат:**

Устанавливает путь к оригинальному тегу для тега-ссылки. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local tag1 = Context:GetNode('/Tags/Project/tag1')
local err = tag1:SetTargetPath('/Project/tag2')
if err ~= nil then print(err) end
```

В примере получаем `'tag1'` из каталога `'Project'` и устанавливаем путь к тегу `'tag2'` из этого же каталога.

## SetTargetTag()

**Синтаксис:**

```
err = tag:SetTargetTag(tag)
```

**Результат:**

Устанавливает оригинальный тег для тега-ссылки. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

## CreateTag()

**Синтаксис:**

```
tag, err = folder:CreateTag(symbolicName, displayName, description, templatePath)
```

**Результат:**

Создает тег. В случае успеха возвращает **nil**, иначе сообщение об ошибке в err.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `symbolicName` | String | Символьное имя тега |
| `displayName` | String | Отображаемое имя тега |
| `description` | String | Описание тега |
| `templatePath` | String | Полный путь к шаблону тега |

**Пример использования:**

```
local source = Context:GetFolder('/Tags/Project')
if source ~= nil then
   for i = 1,10,1 do
      local tag, err = source:CreateTag('tag'..i,'Пример'..i,'','/Templates/tInt')
      if err ~= nil then print(err) end
   end
end
```

В примере получаем папку `'Project;` из корневого каталога `Tags` и создаем в цикле десять нумерованных тегов с символьным именем `'tag'` + порядковый номер, отображаемым именем `'Пример'` + порядковый номер, пустым описанием, используя шаблон `'tInt'` из корневого каталога `Templates`.

## CreateTags()

**Синтаксис:**

```
CreateTags('SymbolicName', 1, n, 'TagTemplate')
```

**Результат:**

Создает теги с нумерованным символьным наименованием `SymbolicName`, с использованием указанного шаблона тега.

**Пример использования:**

```
local t = Context:GetFolder('/Tags/test123/folder1')
t:CreateTags('tag', 1, 50, '/Templates/tInt')
```

В примере в локальную переменную `t` получаем из корневого каталога `Tags` папку `folder1` и в полученной папке создаем 50 нумерованных тегов с символьным именем `tag` + порядковый номер. Все созаднные теги имеют тип `Int`, так как созданы с испольщование шаблона тега `tInt`.

## SetSymbolicName()

**Синтаксис:**

```
err = source:SetSymbolicName(newName)
```

**Результат:**

Устанавливает символьное наименование `newName` серверному объекту в переменной `source`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/Project/OuterValue')
if tag ~= nil then
   local err = tag:SetSymbolicName('InnerValue')
   if err ~= nil then print(err) end
end
```

В примере получаем тег с символьным именем `'OuterValue'` и устанавливаем символьное имя `'InnerValue'`.

## SetDisplayName()

**Синтаксис:**

```
err = source:SetDisplayName(newName)
```

**Результат:**

Устанавливает отображаемое имя `newName` серверному объекту `source`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/Project')
if folder ~= nil then
   local err = folder:SetDisplayName('ASDUE Project')
   if err ~= nil then print(err) end
end
```

В примере получаем папку с символьным именем `'Project'` и устанавливаем отображаемое имя `'ASDUE Project'`.

## SetDescription()

**Синтаксис:**

```
err = source:SetDescription(newDescription)
```

**Результат:**

Устанавливает описание `newDescription` серверному объекту `source`. В случае успеха записывает отображаемое имя объекту и возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local folder = Context:GetFolder('/Tags')
local err = folder:SetDescription('Корневая папка с тегами')
if err ~= nil then print(err) end
```

В примере получаем корневой каталог `Tags` и устанавливаем описание `'Корневая папка с тегами'`.

## SetValue(value)

**Синтаксис:**

```
err = tag:SetValue(value)
```

**Результат:**

Устанавливает значение `value` со статусом `Good` и отметкой времени равной текущему времени тегу `tag`. В случае успеха в `err` возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
function SetTagValue()
   local node = Nodes.TextNode1
   local tag, err = Client:GetNode('/Tags/tag1')
   err = tag:SetValue(100)
   if err ~= nil then node.Text = err end
end
```

В примере при вызове метода мнемосхемы `SetTagValue()` получаем элемент мнемосхемы типа `TextNode` и тег. Затем записываем в тег значение `100`. В случае ошибки записываем сообщение об ошибке в свойство объекта мнемосхемы.

## SetValue(value, statusCode)

**Синтаксис:**

```
err = tag:SetValue(value, statusCode)
```

**Результат:**

Устанавливает значение `value` со статусом `statusCode` тегу `tag`. В случае успеха в `err` возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
function SetTagValue()
   local node = Nodes.TextNode1
   local tag, err = Client:GetNode('/Tags/tag1')
   err = tag:SetValue(100, 0)
   if err ~= nil then node.Text = err end
end
```

В примере при вызове метода мнемосхемы `SetTagValue` получаем элемент мнемосхемы типа `TextNode` и тег. Затем записываем в тег значение `100` со статусом `0` (`Good`). В случае ошибки записываем сообщение об ошибке в свойство объекта мнемосхемы.

## SetValue(value, statusCode, timestamp)

**Синтаксис:**

```
err = tag:SetValue(value, statusCode, timestamp)
```

**Результат:**

Устанавливает значение `value` со статусом `statusCode` и отметкой времени `timestamp` тегу `tag`. В случае успеха в `err` возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
function SetTagValue()
   local node = Nodes.TextNode1
   local tag, err = Client:GetNode('/Tags/tag1')
   local time = os.time({ year = 2017, month = 1, day = 1, hour = 0, min = 0, sec = 0})
   err = tag:SetValue(100, 0, time)
   if err ~= nil then node.Text = err end
end
```

В примере при вызове метода мнемосхемы `SetTagValue` получаем элемент мнемосхемы типа `TextNode` и тег. Затем записываем в тег значение `100` со статусом `0` (`Good`) и отметкой времени `2017-01-01 00:00:00`. В случае ошибки записываем сообщение об ошибке в свойство объекта мнемосхемы.

## ReadHistory()

**Синтаксис:**

```
data, err = tag:ReadHistory(startTicks, endTicks, bounds)
```

До версии 3.3.19:

```
err, data = tag:ReadHistory(startTicks, endTicks, bounds)
```

**Результат:**

Возвращает в `data` историю значений тега `tag` за период с `startTicks` до `endTicks` и **nil** в `err`. Логическое поле `bounds` показывает, включать граничные значения в запрос истории или нет. В случае ошибки возвращает сообщение об ошибке в `err`.

Каждый объект массива `data` содержит следующие свойства:

| Имя | Тип | Описание |
|---|---|---|
| `Value` | Object | Значение тега |
| `HasValue` | Boolean | **true**, если значение тега не равно **nil**, иначе **false** |
| `ValueAsInt` | Number | значение тега, приведенное к типу Int32 |
| `ValueAsLong` | Number | значение тега, приведенное к типу Int64 |
| `ValueAsString` | String | значение тега, приведенное к типу String |
| `ValueAsDouble` | Double | значение тега, приведенное к типу Double |
| `StatusCode` | Number | код статуса значения |
| `ServerTimestamp` | DateTime | отметка времени получения сервером значения |
| `SourceTimestamp` | DateTime | отметка времени возникновения значения |
| `SourceTicks` | Number | отметка времени получения сервером значения в тиках |
| `ServerTicks` | Number | отметка времени возникновения значения в тиках |

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `startTicks` | Number | Начало временного интервала |

| Имя | Тип | Описание |
|---|---|---|
| `endTicks` | Number | Конец временного интервала |
| `bounds` | Boolean | Флаг включения граничных значений |

**Пример использования:**

```
--количество тиков в секунде
local ticks = 10 ^ 7
--интервал в тиках равен 1 час * 60 минут * 60 секунд * тиков в секунду
local interval = 1 * 60 * 60 * ticks
local endTicks = os.time()
local startTicks = os.time()-interval
local tag = Context:GetNode('/Tags/tag1')
local values, err = tag:ReadHistory(startTicks, endTicks, true)
if err ~= nil then print(err) return end
local file = io.open( 'C:/Dev/file.txt ', 'w ')
for i = 0, values.Length-1 do
   if values[i].HasValue then
      file:write(tostring(values[i].Value).. '\t ')
      file:write(values[i].StatusCode.. '\t ')
      file:write(values[i].ServerTimestamp:ToString().. '\t ')
      file:write(values[i].SourceTimestamp:ToString().. '\t ')
      file:write(values[i].ServerTicks.. '\t ')
      file:write(values[i].SourceTicks.. '\r\n ')
   end
end
file:close()
```

В примере читаем за последний час значения тега с символьным именем `'tag1'` из корневого каталога `Tags` и записываем в файл значение тега, его код статуса, отметку времени сервера и источника, а также отметку времени сервера и источника в тиках, в конце каждой итерации переходим на новую строку.

> ***Прим.:*** *т.к. в значении `Value` может быть не только строка, но и **nil** или число, то используем стандартную lua функцию приведения к строковому типу `tostring()`, в то время как для объектов типа `userdata` поддерживается метод приведения к строке `ToString()`.*

## ReadProcessedHistory()

**Синтаксис:**

```
data, err = tag:ReadProcessedHistory(startTicks, endTicks, aggregateName, interval,
 uncertainAsBad, percentBad, percentGood, slopedExtrapolation, steppedInterpolation)
```

До версии 3.3.19:

```
err, data = tag:ReadProcessedHistory(startTicks, endTicks, aggregateName, interval,
 uncertainAsBad, percentBad, percentGood, slopedExtrapolation, steppedInterpolation)
```

**Результат:**

Возвращает в `data` агрегированные функцией `aggregateName` значения тега `tag` за период от `startTicks` до `endTicks` с интервалом агрегирования `interval`. В случае ошибки возвращает сообщение об ошибке в `err`. Подробную информацию по работе с функциями агрегирования см. в руководстве по созданию серверных объектов.

Элементы коллекции `data` содержат такие же свойства, что и объекты при вызове функции *ReadHistory()*.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `startTicks` | Number | Начало временного интервала |
| `endTicks` | Number | Конец временного интервала |
| `aggregateName` | String | наименование функции |
| `interval` | | интервал агрегирования |
| `uncertainAsBad` | Boolean | логическое значение, если **true**, то значения со статусом Uncertain при вычислении не учитываются |
| `percentBad` | Number | допустимое количество плохих значений, в процентах |
| `percentGood` | Number | допустимое количество качественных значений, в процентах |
| `slopedExtrapolation` | Boolean | логическое значение, если **true**, то применяется линейная экстраполяция, если **false** - ступенчатая экстраполяция |
| `steppedInterpolation` | Boolean | логическое значение, если **true**, то применяется ступенчатая интерполяция, если **false** - линейная интерполяция |

**Пример использования:**

```
--количество тиков в секунде
local ticks = 10 ^ 7
--интервал в тиках равен 1 час * 60 минут * 60 секунд * тиков в секунду
local interval = 1 * 60 * 60 * ticks
local endTicks = os.time()
local startTicks = os.time()-interval
local tag = Context:GetNode('/Tags/tag1')
local   data,   err   =   tag:ReadProcessedHistory(startTicks,   endTicks,   'Average',
 interval, true, 100, 0, false, true)
if err ~= nil then print(err) return end
local file = io.open('C:/Dev/file.txt','w')
for i = 0, data.Length-1 do
   if data[i].HasValue then
      file:write(tostring(data[i].Value)..'\t')
      file:write(data[i].StatusCode..'\t')
      file:write(data[i].ServerTimestamp:ToString()..'\t')
      file:write(data[i].SourceTimestamp:ToString()..'\t')
      file:write(data[i].ServerTicks..'\t')
      file:write(data[i].SourceTicks..'\r\n')
   end
end
file:close()
```

В примере получаем агрегированное значение тега с символьным именем `tag1` за последний час, используя агрегат `Average` и записываем в файл полученное значение, его код статуса, отметку времени сервера и источника, а также отметку времени сервера и источника в тиках, в конце каждой итерации переходим на новую строку. Так как мы указали интервал `interval` равный периоду, то агрегированное значение у нас будет одно.

> ***Прим.:*** *т.к. в значении `Value` в нашем примере будет число, то используем стандартную lua функцию приведения к строковому типу `tostring()`, в то время как для объектов типа `userdata` поддерживается метод приведения к строке `ToString()`.*

# 11. Управление шаблонами тегов

## 11.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *External* | Boolean | Возвращает значение свойства (хранение значения тега в файловой системе без хранения истории значений). Если возвращает **true**, значит тег, созданный по данному шаблону, хранится в базе данных в папке external. Если **false**, в папке items. |
| *SetExternal()* | | Поволяет присвоить значение свойству **External**. |
| *TagValueType* | String | Возвращает тип значения тега. |
| *Archiving-Options.Depth* | String | Возвращает значение глубины хранения истории значений (недоступно для типов ByteArray и String с хранением значения в файловой системе). |
| *SetDepth()* | | Поволяет присвоить значение свойству **Depth** (блок свойств **ArchivingOptions**). |
| *Archiving-Options.Deadband-Value* | Double | Возвращает значение зоны нечувствительности (доступно только для типов Int и Double). |
| *SetDeadBand()* | | Поволяет присвоить значение свойству **Deadband** (блок свойств **ArchivingOptions**). |
| *Archiving-Options.ReadCache-Size* | Int | Возвращает количество точек, сохраняемых в кэше. |
| *SetReadCacheSize()* | | Поволяет присвоить значение свойству **ReadCacheSize** (блок свойств **ArchivingOptions**). |
| *Archiving-Options.Tmax* | Int | Возвращает значение промежутка времени, по истечении которого новое значение обязательно сохраняется. |

| Имя | Тип | Описание |
|---|---|---|
| SetTmax() | | Поволяет присвоить значение свойству **Tmax** (блок свойств **ArchivingOptions**). |
| Archiving-Options.Tmin | Int | Возвращает значение промежутка времени, в течение которого новые значения отбрасываются. |
| SetTmin() | | Поволяет присвоить значение свойству **Tmin** (блок свойств **ArchivingOptions**). |
| LimitingOptions.Type | String | Возвращает тип ограничения диапазона значений. |
| LimitingOptions.High | Double | Возвращает значение верхней границы допустимых значений. |
| LimitingOptions.Low | Double | Возращает значение нижней границы допустимых значений. |
| SetLimitingTypeAndBounds() | | Позволяет присвоить значения свойствам блока **Limiting Options**. |

## External

**Синтаксис:**

```
template.External
```

**Результат:**

Возвращает значение свойства.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.External)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **External**, заданное шаблону тега при его создании: `true`, если флаг свойства **External Storage** установлен, иначе - `false`.

## SetExternal()

**Синтаксис:**

```
template:SetExternal()
```

**Результат:**

Поволяет присвоить значение свойству **External**.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tString')
template:SetExternal(true)
print(template.External)
```

В примере получаем шаблон тега типа **String** и присваиваем свойсвту **External** значение true.

## TagValueType

**Синтаксис:**

```
template.TagValueType
```

**Результат:**

Возращает тип значения тега.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.TagValueType)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **TagValurType**, заданное шаблону тега при его создании.

## ArchivingOptions.Depth

**Синтаксис:**

```
template.ArchivingOptions.Depth
```

**Результат:**

Возращает значение глубины хранения истории значений.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.ArchivingOptions.Depth)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Depth** (блок свойств **ArchivingOptions**), заданное шаблону тега при его создании.

## SetDepth()

**Синтаксис:**

```
template:SetDepth()
```

**Результат:**

Поволяет присвоить значение свойству **Depth** (блок свойств **ArchivingOptions**).

**Пример использования:**

```
local template = Context:GetNode('/Templates/tString')
template:SetDepth(13)
print(template.ArchivingOptions.Depth)
```

В примере получаем шаблон тега типа **String** и присваиваем свойству **Depth** (блок свойств **ArchivingOptions**) значение 13, что соответствует позиции выпадающего списка - 1 year 1 month.

## ArchivingOptions.DeadbandValue

**Синтаксис:**

```
template.ArchivingOptions.DeadbandValue
```

**Результат:**

Возращает значение зоны нечувствительности.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.ArchivingOptions.DeadbandValue)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Deadband** (блок свойств **ArchivingOptions**), заданное шаблону тега при его создании.

## SetDeadBand()

**Синтаксис:**

```
template:SetDeadBand()
```

**Результат:**

Поволяет присвоить значение свойству **Deadband** (блок свойств **ArchivingOptions**).

**Пример использования:**

```
local template = Context:GetNode('/Templates/tDouble')
template:SetDeadBand(14.01)
print(template.ArchivingOptions.DeadbandValue)
```

В примере получаем шаблон тега типа **Double** и присваиваем свойству **Deadband** значение 14.01.

## ArchivingOptions.ReadCacheSize

**Синтаксис:**

```
template.ArchivingOptions.ReadCacheSize
```

**Результат:**

Возращает количество точек, сохраняемых в кэше.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.ArchivingOptions.ReadCacheSize)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **ReadCacheSize** (блок свойств **ArchivingOptions**), заданное шаблону тега при его создании.

## SetReadCacheSize()

**Синтаксис:**

```
template:SetReadCacheSize()
```

**Результат:**

Поволяет присвоить значение свойству **Read cache size** (блок свойств **ArchivingOptions**)..

**Пример использования:**

```
local template = Context:GetNode('/Templates/tDouble')

template:SetReadCacheSize(10)

print(template.ArchivingOptions.ReadCacheSize)
```

В примере получаем шаблон тега типа **Double** и присваиваем свойству **Read cache size** значение 10.

## ArchivingOptions.Tmax

**Синтаксис:**

```
template.ArchivingOptions.Tmax
```

**Результат:**

Возращает промежуток времени, по истечении которого новое значение обязательно сохраняется.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.ArchivingOptions.Tmax)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Tmax** (блок свойств **ArchivingOptions**), заданное шаблону тега при его создании.

## SetTmax()

**Синтаксис:**

```
template:SetTmax()
```

**Результат:**

Поволяет присвоить значение свойству **Tmax** (блок свойств **ArchivingOptions**)..

**Пример использования:**

```
local template = Context:GetNode('/Templates/tDouble')

template:SetTmax(50)

print(template.ArchivingOptions.Tmax)
```

В примере получаем шаблон тега типа **Double** и присваиваем свойству **Tmax** значение 50.

## ArchivingOptions.Tmin

**Синтаксис:**

```
template.ArchivingOptions.Tmin
```

**Результат:**

Возращает промежуток времени, в течение которого новые значения отбрасываются.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.ArchivingOptions.Tmin)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Tmin** (блок свойств **ArchivingOptions**), заданное шаблону тега при его создании.

## SetTmin()

**Синтаксис:**

```
template:SetTmin()
```

**Результат:**

Поволяет присвоить значение свойству **Tmin** (блок свойств **ArchivingOptions**).

**Пример использования:**

```
local template = Context:GetNode('/Templates/tDouble')

template:SetTmin(20)

print(template.ArchivingOptions.Tmin)
```

В примере получаем шаблон тега типа **Double** и присваиваем свойству **Tmin** значение 20.

## LimitingOptions.Type

**Синтаксис:**

```
template.LimitingOptions.Type
```

**Результат:**

Возвращает тип ограничения диапазона значений.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.LimitingOptions.Type)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Type** (блок свойств **LimitingOptions**), заданное шаблону тега при его создании.

## LimitingOptions.High

**Синтаксис:**

```
template.LimitingOptions.High
```

**Результат:**

Возвращает значение верхней границы допустимых значений.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.LimitingOptions.High)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **High** (блок свойств **LimitingOptions**), заданное шаблону тега при его создании.

## LimitingOptions.Low

**Синтаксис:**

```
template.LimitingOptions.Low
```

**Результат:**

Возвращает значение нижней границы допустимых значений.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')
print(template.LimitingOptions.Low)
```

В примере получаем шаблон тега типа **Int** и выводим на печать значение свойства **Low** (блок свойств **LimitingOptions**), заданное шаблону тега при его создании.

## SetLimitingTypeAndBounds()

**Синтаксис:**

```
template:SetLimitingTypeAndBounds(type, low, high)
```

**Результат:**

Позволяет присвоить значения свойствам блока **Limiting Options**.

**Пример использования:**

```
--
local template = Context:GetNode('/Templates/tDouble')
template:SetLimitingTypeAndBounds('HighLow')
template:SetLow(8)
template:SetHigh(10)
print(template.LimitingOptions.Type)
print(template.LimitingOptions.Low)
print(template.LimitingOptions.High)


--
local template = Context:GetNode('/Templates/tDouble')
template:SetLimitingTypeAndBounds('HighLow',12,13)
print(template.LimitingOptions)
```

В примере получаем шаблон тега типа **Double** и присваиваем свойствам блока **Limiting Options** значения.

# 11.2. Методы

**Методы:**

| Метод | Описание |
|---|---|
| *CreateTagTemplate()* | Создает шаблон тега. |
| *GetAllTagsUseTheTemplate()* | Позволяет получить список тегов, использующих выбранный шаблон. |
| *CheckCreateTemplate()* | Проверяет БД на наличие указанного шаблона тега, в случае его отсутствия создает его. |
| *Delete()* | Позволяет удалить шабон тега, но только в том случае если по данному шаблону не созданы теги. |

## CreateTagTemplate()

**Синтаксис:**

```
local tag, err = folder:CreateTagTemplate(symbolicName, displayName, description,
 tagValueType)
```

**Результат:**

Создает шаблон тега. В случае успеха возвращает nil, иначе сообщение об ошибке в err.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `SymbolicName` | String | Возвращает символьное имя шаблона |
| `DisplayName` | String | Возвращает отображаемое имя шаблона |
| `Description` | String | Создает описание шаблона |
| `TagValueType` | String | Возвращает тип шаблона тега |

**Пример использования:** создание шаблонов / шаблона тега

```
local folder, err = Context:GetFolder('/Templates')
if err ~= nil then print(err) end


if folder ~= nil then
local subFolder, err =
 folder:CreateTagTemplate('SymbName','DispName','Description','Int')
if err ~= nil then print(er) end
end
-----------------------------------------------------------------
local folder, err = Context:GetFolder('/Templates/createdTemplates')
if err~= nil then print(err) end
  for i = 1,5,1 do
   local tag, err = folder:CreateTagTemplate('template'..i, 'template'..i, 'desc', 'Int')
    if err ~= nill then print(err) end
end
```

## GetAllTagsUseTheTemplate()

**Синтаксис:**

```
local tags = template:GetAllTagsUseTheTemplate()
```

**Результат:**

Позволяет получить список тегов, использующих выбранный шаблон.

**Пример использования:**

```
local template = Context:GetNode('/Templates/tInt')


local tags = template:GetAllTagsUseTheTemplate()
  if tags ~= nil then
    for i = 0, tags.Length - 1 do
    print(tags[i].SymbolicName)
  end
end
```

В примере получаем папку с шаблоном тега, затем выводим список всех тегов, в которых при создании был указан данный шаблон тега.

Проверить работоспособность метода можно, запустив сервер из под консоли: лаунчер → Console server.

## CheckCreateTemplate()

**Синтаксис:**

```
local t = CheckCreateTemplate(templatesRoot,'Workstation','Шаблон настроек APMa',0,12,0,0,0,0,0)
```

**Результат:**

Проверяет БД на наличие указанного шаблона тега, в случае его отсутствия создает его.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `ParentFolder` | String | Родительская папка |
| `SymbolicName` | String | Символьное имя |
| `Description` | String | Описание |
| `ValueType` | Int | Тип значения:<br>• 0-string,<br>• 1-int,<br>• 2-long,<br>• 3-double,<br>• 4-byteArray |
| `Depth` | Int | Глубина архива |
| `Deadband` | Int | Погрешность архива |
| `CacheSize` | Int | Количество точек, сохраняемых в кэше. |
| `LimitType` | Int | Тип лимита:<br>• 0-none,<br>• 1-hi,<br>• 2-lo,<br>• 3-hilo |
| `Min` | Int | Минимальный промежуток времени, в течение которого новые значения отбрасываются. |

| Имя | Тип | Описание |
|---|---|---|
| Max | Int | Максимальный промежуток времени, по истечении которого новое значение обязательно сохраняется. |

**Пример использования:**

В качестве примера приведено описание функции **CheckCreateTemplate()** из *heatingUtils.lua* и пример практического использования этой же функции в *initHeating.lua*:

```
-- heatingUtils.lua
function CheckCreateTemplate(parentFolder, symbolicName, desc, valueType, depth,
 deadband, cacheSize, limitType, min, max)
local t = Browser:BrowseNode(symbolicName, parentFolder)
if t == nil then
print('Creating template '..symbolicName)
t = Nodes.TemplateNode()
t.SymbolicName = symbolicName
t.DisplayName = symbolicName
t.Description = desc
t.TagValueType = TagValueTypes[valueType]
t.ArchivingOptions.Depth = depth
t.ArchivingOptions.DeadbandValue = deadband
t.ArchivingOptions.ReadCacheSize = cacheSize
t.LimitingOptions.Type = RangeTypes[limitType]
t.LimitingOptions.High = max
t.LimitingOptions.Low = min
t:Create(Browser.Connection, parentFolder)
end
return t
end


-- initHeating.lua
local templatesRoot = CheckCreateFolder(root, prjName)
local t
t = CheckCreateTemplate(templatesRoot,'Workstation','Шаблон настроек
 APMa',0,12,0,0,0,0,0)
t = CheckCreateTemplate(templatesRoot,'WorkstationActive','Шаблон активности
 APMa',2,0,0,0,0,0,0)
```

```
t = CheckCreateTemplate(templatesRoot,'WorkstationHeartbeatPeriod','Шаблон периода записи, в секундах (1..600)',1,0,0,0,0,0,0)
t = CheckCreateTemplate(templatesRoot,'Factory','Шаблон имени завода',0,0,0,0,0,0,0)
```

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

В случае успеха удаляет шаблон тега и возвращает nil.

**Пример использования:**

```
local tmp = Context:GetNode('/Templates/tString')
tmp:Delete()
```

В примере в локальную перменную `tmp` получаем шаблон тега `tString` из корневого каталога `Templates`, затем удаляем шаблон тега.

# 12. Управление джобами

В данном разделе рассматривается пример работы с **серверными переменными**. Серверные переменные представляют собой таблицу, используются для оперативного доступа к данным из любого джоба, но при соблюдении условий:

- серверная переменная и джоб находятся на одном сервере,
- сервер активен.

Джоб 1 [объявляем серверные переменные]:

```
Context:GlobalSet('varA', 2)
Context:GlobalSet('varB', 4)
```

Джоб 2 [используем, ранее созданные в джобе 1, серверные переменные и выводим результат на печать]

```
local a = Context:GlobalGet('varA')
local b = Context:GlobalSet('varB')
print(a+b)
```

Для очистки таблицы серверных переменных используйте Global:Clear()

# 12.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *SetScript()* | | Позволяет записать любой скрипт в тело джоба. |
| *SetStartup()* | | Позволяет задать тип запуска джоба. |
| *SetTemplatePaths()* | | Позволяет задать джобу с типом запуска **TagValueChange** путь к шабонам тегов. |
| *SetTagPaths()* | | Позволяет задать джобу с типом запуска **TagValueChange** путь к тегам. |
| *SetRunAs()* | | Позволяет задать Пользователя, от имени которого должен быть запущен джоб. |
| *SetSchedule()* | | Позволяет задать расписание, по которому будет запускаться джоб. |
| *Status* | | Позволяет получить статус джоба. |
| *SetWatchdogStatus()* | | Позволяет вкл / выкл таймер. |
| *SetWatchdogTimer()* | | Позволяет задать период срабатывания таймера в секундах. |

## SetScript()

**Синтаксис:**

```
job:SetScript([[code]])
```

**Результат:**

Позволяет записать любой скрипт в тело джоба.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/jobSymbName')
if job == nil then print('Not found job') end
job:SetScript([[
local folder = Context:GetFolder('/Jobs')
folder:CreateJob('jobSymbName_1', 'jobDispName', 'jobDesc', 'Manual', 'admin')
]])
```

В примере получаем джоб **jobSymbName** из корневой папки **Jobs**, проверяем есть ли такой джоб, затем записываем в тело джоба скрипт по созданию нового джоба **jobSymbName_1**.

## SetStartup()

**Синтаксис:**

```
job:SetStartup(startup)
```

**Результат:**

Позволяет задать тип запуска джоба.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/jobSymbName')
job:SetStartup('disabled')
```

В примере получаем джоб **jobSymbName** из корневой папки **Jobs**, затем задаем тип запуска этому джобу **Disabled**.

## SetTemplatePaths() и SetTagPaths()

**Синтаксис:**

```
-- задать пути шаблонов тегов
job:SetTemplatePaths(templatePaths)


-- задать пути тегов
job:SetTagPaths(tagPaths)
```

**Результат:**

Позволяет задать джобу с типом запуска **TagValueChange** пути шаблонов либо тегов. В этом случае джоб, с заданным свойством **TagValueChange**, будет запускаться всякий раз, когда будут меняться значения тега / тегов, которые были созданы по указанным шаблонам.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/jobSymbName')
local templateFolder = Context:GetFolder('/Templates/ASDUE/Workstation')
local templates = templateFolder:GetNodes()
local templatePaths = {}
for i = 0, templates.Length-1 do
 templatePaths[i] = templates[i].Path
end
error = job:SetTemplatePaths(templatePaths)
if error ~= nil then print(error) end
local jobTemplatePaths = job.TemplatePaths
for i = 0, templates.Length-1 do
 print(job.TemplatePaths[i])
end
```

В примере для указанного джоба с типом запуска **TagValueChange** задаем все шаблоны тегов, которые расположены в папке */Templates/ASDUE/Workstation*.

```
local job = Context:GetNode('/Jobs/jobFolder/job')
local tagFolder = Context:GetFolder('/Tags/dem')
local tags = tagFolder:GetNodes()
local tagPaths = {}
for i = 0, tags.Length-1 do
        tagPaths[i] = tags[i].Path
end
```

```
error = job:SetTagPaths(tagPaths)

if error ~= nil then print(error) end

local jobTagPaths = job.TagPaths

for i = 0, tags.Length-1 do

        print(job.TagPaths[i])

end
```

В примере для указанного джоба с типом запуска **TagValueChange** задаем все теги, которые расположены в папке */Tags/dem*.

## SetRunAs()

**Синтаксис:**

```
job:SetRunAs('user_name')
```

**Результат:**

Позволяет указать от имени какого Пользователя должен запускаться джоб.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/job2')
job:SetRunAs('AdminDemo')
```

В примере получаем джоб **job2** из корневой папки **Jobs**, затем указываем Пользователя, от имени которого должен запускаться джоб.

## SetSchedule()

**Синтаксис:**

```
job:SetSchedule('schedule')
```

**Результат:**

Позволяет задать расписание, по которому будет запускаться джоб.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/job2')
job:SetSchedule('*/30 * * * * *')
```

В примере получаем джоб **job2** из корневой папки **Jobs**, затем указываем расписание запуска джоба (в данном случае каждые 30 секунд).

## Status

**Синтаксис:**

```
job.Status
```

**Результат:**

Позволяет получить статус джоба.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/job2')

print(job.Status)
```

В примере получаем джоб **job2** из корневой папки **Jobs**, затем получаем статус джоба.

## SetWatchdogStatus()

**Синтаксис:**

```
job:SetWatchdogStatus(true)
```

**Результат:**

Позволяет вкл / выкл таймер.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/generate')

job:SetWatchdogStatus(true)
```

В примере получаем джоб **generate** из корневой папки **Jobs**, затем включаем таймер джоба.

## SetWatchdogTimer()

**Синтаксис:**

```
job:SetWatchdogTimer(7)
```

**Результат:**

Позволяет задать период срабатывания таймера в секундах.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/generate')
job:SetWatchdogStatus(true)
job:SetWatchdogTimer(7)
```

В примере получаем джоб **generate** из корневой папки **Jobs**, затем включаем таймер джоба и указываем период срабатывания таймера.

# 12.2. Методы

**Методы:**

| Метод | Описание |
|---|---|
| *CreateJob()* | Позволяет создать джоб. |
| *CreateJobFolder()* | Позволяет создать папку для джобов. |
| *Delete()* | Позволяет удалить джоб. |

## CreateJob()

**Синтаксис:**

```
folder:CreateJob('jobSymbName', 'jobDispName', 'jobDesc', 'start up', 'run as')
```

**Результат:**

Позволяет создать джоб и указать: тип запуска и Пользователя, от имени которого должен запускаться джоб.

**Пример использования:**

```
local folder = Context:GetFolder('/Jobs')
folder:CreateJob('jobSymbName', 'jobDispName', 'jobDesc', 'Manual', 'admin')
```

В примере создаем папку с джобами, в которой будет создан джоб, затем создаем джоб, указав типа запуска Manual и Пользователя - admin.

## CreateJobFolder()

**Синтаксис:**

```
folder:CreateJobFolder('jobFolderSymbName','jobFolderDispName','jobFolderDesc')
```

**Результат:**

Позволяет создать папку для джобов.

**Пример использования:**

```
------
local folder = Context:GetFolder('/Jobs')
folder:CreateJobFolder('jobSymbName_','jobDispName_',' ')
------
local folder = Context:GetFolder('/Jobs')
if folder ~= nil then
  for i = 1,3,1 do
  local subFolder, er = folder:CreateJobFolder('jobSymbName'..i,'
  jobDispName'..i,' ')
    if er ~= nil then print(er) end
  end
end
```

В примере создаем папку / папки в корневой папке Jobs.

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

В случае успеха удаляет джоб и возвращает nil, иначе возвращает сообщение об ошибке.

**Пример использования:**

```
local job = Context:GetNode('/Jobs/job2')
job:Delete()
```

В примере в локальную перменную `job` получаем джоб `job2` из корневого каталога `Jobs`, затем удаляем джоб.

# 13. Управление схемой данных и отчетом

В данном разделе перечислены методы и свойства для подготовки схемы данных, а также методы по созданию и сохранению отчета по шаблону, разработанному в среде Report Designer.

## 13.1. Свойства

| Имя | Тип | Описание |
|---|---|---|
| *EnforceConstraints* | Boolean | Возвращает и устанавливает состояние проверки ограничений при добавлении записей |

## EnforceConstraints

**Синтаксис:**

```
flag = dataSet.EnforceConstraints
```

**Результат:**

Возвращает состояние проверки ограничений при добавлении записей. Установите значение **false**, чтобы отключить проверку. По умолчанию значение **true**. Обычно проверку отключают на этапе добавления большого количества записей. После завершения операции добавления проверку ограничений принято включать заново.

**Пример использования:** см. в описании метода *CreateDataSet()*.

# 13.2. Методы

| Имя | Описание |
|---|---|
| *CreateDataSet(name)* | Возвращает пустую схему данных |
| *AddTable(name, columns, primaryKey)* | Добавляет в схему данных таблицу с заданными колонками и ключом |
| *AddRelation(name, parentTable, parentColumn, childTable, childColumn)* | Добавляет в схему данных связь между таблицами |
| *AddRow(name, value1, value2, ...)* | Добавляет строку в таблицу с указанными значениями |
| *Save(tag)* | Сохраняет схему данных в тег типа string |
| *RemoveRelation(name)* | Удаляет указанную связь из схемы данных |
| *RemoveTable(name)* | Удаляет указанную таблицу из схемы данных |
| *CreateReport(template, dataSet)* | Возвращает отчет по указанным шаблону отчета и схеме данных |
| *SetParameter(name, value)* | Устанавливает значение параметру отчета |
| *GetParameter(name)* | Возвращает значение параметра отчета |
| *ExportToTag(format, tag)* | Сохраняет отчет в тег типа ByteArray |
| *ExportToTag(format, tag, useTimeout)* | Сохраняет отчет в тег типа ByteArray |
| *ExportToFile(format, fullPath)* | Сохраняет отчет в файл |
| *ExportToFile(format, fullPath, useTimeout)* | Сохраняет отчет в файл |
| *Print(printerName)* | Отправляет отчет на печать |

## CreateDataSet()

**Синтаксис:**

```
dataSet = Context:CreateDataSet(dataSetName)
```

**Результат:**

Возвращает в переменную `dataSet` пустую схему с именем `dataSetName`.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/Runtime/Reports/dataset')
if tag == nil then error('DataSet tag not found') end
--создаем схему данных
local dataSet = Context:CreateDataSet('SimpleDataSet')
local err
--добавляем таблицы в схему
err = dataSet:AddTable('Boxes', 'Id,BoxName,Description', 'Id')
if err ~= nil then error(err) end
err = dataSet:AddTable('Products', 'Id,ProductName,Count,Color,BoxId', 'Id')
if err ~= nil then error(err) end
--добавляем связь между таблицами
err = dataSet:AddRelation('ProductBox', 'Boxes', 'Id', 'Products', 'BoxId')
if err ~= nil then error(err) end
dataSet.EnforceConstraints = false
--заполняем таблицы данными
err = dataSet:AddRow('Boxes', 1, 'Simple Box', 'Wooden box')
if err ~= nil then error(err) end
err = dataSet:AddRow('Products', 1, 'Apples', 100, 'Green', 1)
if err ~= nil then error(err) end
dataSet.EnforceConstraints = true
--сохраняем схему данных в строковый тег
err = dataSet:Save(tag)
if err ~= nil then error(err) end
--получаем тег типа ByteArray с шаблоном отчета
local template = Context:GetNode('/Tags/Runtime/Reports/template')
if template == nil then error('Report template not found') end
err, report = Context:CreateReport(template, dataSet)
if err ~= nil then error(err) end
err = report:SetParameter('Customer', 'Oil Plant')
```

```
if err ~= nil then error(err) end
local reportTag = Context:GetNode('/Tags/Runtime/Reports/Exported/report')
if reportTag == nil then error('Report tag not found') end
--получаем параметр из отчета с именем Supplier
--строковый параметр Supplier содержится в шаблоне отчета
local parameter, err = report:GetParameter('Supplier')
if err ~= nil then error(err) end
reportTag:SetDisplayName(parameter)
err = report:ExportToTag('pdf', reportTag)
if err ~= nil then error(err) end
err = report:ExportToFile('pdf', 'C:/Dev/reportExample.pdf')
if err ~= nil then error(err) end
err = report:Print('HP LaserJet 1020')
if err ~= nil then print(err) end
err = dataSet:RemoveRelation('ProductBox')
if err ~= nil then error(err) end
```

В примере создаем схему данных, таблицы и связь между ними, заполняем таблицы данными и сохраняем схему в строковый тег. Создаем отчет по схеме данных и шаблону отчета и сохраняем в тег, в файл, отправляем на печать. После выполнения всех функций удаляем связь из схемы данных.

## AddTable()

**Синтаксис:**

```
err = dataSet:AddTable(name, columns, primaryKey)
```

**Результат:**

Добавляет в схему `dataSet` таблицу с именем `name` со столбцами из `columns` и главным ключом `primaryKey` (может быть пустым). В случае успеха возвращает **nil**, иначе сообщение об ошибке.

> ***Прим.:*** *для корректного сохранения названий столбцов в схеме документа рекомендуется при перечислении столбцов вторым аргументом не ставить пробелы после запятой. Пример использования см. в описании метода CreateDataSet().*

## AddRelation()

**Синтаксис:**

```
err = dataSet:AddRelation(name, parentTableName, parentColumnName, childTableName, childColumnName)
```

**Результат:**

Добавляет в схему `dataSet` связь с именем `relationName` между таблицей `parentTableName` по столбцу `parentColumnName` и таблицей `childTableName` по столбцу `childColumnName`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## AddRow()

**Синтаксис:**

```
err = dataSet:AddRow(name, value1, value2, ...)
```

**Результат:**

Добавляет строку в таблицу `name` со значениями `value1`, `value2`, ... В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## Save()

**Синтаксис:**

```
err = dataSet:Save(tag)
```

**Результат:**

Сохраняет схему `dataSet` в строковый тег `tag`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## RemoveRelation()

**Синтаксис:**

```
err = dataSet:RemoveRelation(name)
```

**Результат:**

Удаляет связь с именем `name` из схемы `dataSet`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## RemoveTable()

**Синтаксис:**

```
err = dataSet:RemoveTable(name)
```

**Результат:**

Удаляет таблицу с именем `name` из схемы `dataSet`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## CreateReport()

**Синтаксис:**

```
report, err = Context:CreateReport(reportTemplate, dataSet)
```

До версии 3.3.19:

```
err, report = Context:CreateReport(reportTemplate, dataSet)
```

**Результат:**

Формирует отчет `report` по шаблону `reportTemplate` и схеме `dataSet`. Параметр `reportTemplate` - тег типа `ByteArray`, в котором хранится шаблон отчета. В случае успеха возвращает сформированный отчет в `report` и **nil** в `err`, иначе сообщение об ошибке в `err` и **nil** в `report`.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## SetParameter()

**Синтаксис:**

```
err = report:SetParameter(parameterName, parameterValue)
```

**Результат:**

Устанавливает значение `parameterValue` параметра с именем `parameterName` отчету `report`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## GetParameter()

**Синтаксис:**

```
parameter, err = report:GetParameter(nameParameter)
```

До версии 3.3.19:

```
err, parameter = report:GetParameter(nameParameter)
```

**Результат:**

Возвращает значение параметра `parameter` с именем `nameParameter` отчета `report`. В случае успеха возвращает **nil** в `err` и значение параметр в `parameter`, иначе сообщение об ошибке в `err` и **nil** в `parameter`. Тип значения `parameter` зависит от значения параметра в шаблоне отчета.

**Пример использования:** см. в описании метода *CreateDataSet()*.

## ExportToTag(format, tag)

**Синтаксис:**

```
err = report:ExportToTag(format, tag)
```

**Результат:**

Сохраняет отчет `report` в формате `format` в тег `tag`. В случае успеха возвращает **nil**, иначе сообщение об ошибке. Поддерживаемые форматы (значения переменной `format` в виде строки): `'csv'`, `'html'`, `'pdf'`, `'rtf'`, `'text'`, `'mht'`, `'xls'`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `format` | String | Формат отчета |
| `tag` | String | Символьное имя тега |

**Пример использования:** см. в описании метода *CreateDataSet()*.

## ExportToTag(format, tag, useTimeout)

**Синтаксис:**

```
err = report:ExportToTag(format, tag, useTimeout)
```

**Результат:**

Сохраняет отчет `report` в формате `format` в тег `tag`. В случае успеха возвращает **nil**, иначе сообщение об ошибке. `useTimeout` - логическое значение, которое позволяет игнорировать таймауты времени исполнения серверного скрипта (если **false**) и используется для выгрузки отчетов с большим объемом данных.

Поддерживаемые форматы (значения переменной `format` в виде строки): `'csv'`, `'html'`, `'pdf'`, `'rtf'`, `'text'`, `'mht'`, `'xls'`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `format` | String | Формат отчета |
| `tag` | String | Символьное имя тега |
| `useTimeout` | Boolean | Логическое значение, которое позволяет игнорировать таймауты времени исполнения серверного скрипта и используется для выгрузки отчетов с большим объемом данных |

**Пример использования:** см. в описании метода *CreateDataSet()*.

> ***Прим.:*** *при экспорте в тег значение не должно быть больше 10 Мб, отчеты больше 10 Мб в тег на сервере сохраняться не будут.*

## ExportToFile(format, fullPath)

**Синтаксис:**

```
err = report:ExportToFile(format, path)
```

**Результат:**

Сохраняет отчет `report` в формате `format` в файл по пути `path`. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

Поддерживаемые форматы (значения переменной `format` в виде строки): `'csv'`, `'html'`, `'pdf'`, `'rtf'`, `'text'`, `'mht'`, `'xls'`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `format` | String | Формат отчета |
| `path` | String | Путь для сохранения отчета |

**Пример использования:**

```
-- Пример для создания локального (хранимого на жестком диске) отчета, с
-- возможностью чтения/отображения во вкладке с отчетами
ReportPath = 'C:/'
err = report:ExportToFile('pdf', ReportPath..fileName..'.pdf')
if err ~= nil and err ~= '' and err~= ' ' then
reportSystem:ReportEvent(reportFolder, 700, err)
end
-- Создание стринг-тега для хранения пути к локальному отчету
t,err = pdfFolder:CreateTag(fileName, displayName, '', projectPath..'/Reports/
templateString')
if err ~= nil and err ~= '' and err~= ' ' then
reportSystem:ReportEvent(reportFolder, 700, err)
end
-- Выставление пути к локальному отчету в Value тега
ReportFile = Context:GetTag(projectPath..'/Reports/
Exported/'..reportFolder..'/'..fileName)
ReportFile:SetValue(ReportPath..fileName..'.pdf')
```

## ExportToFile(format, fullPath, useTimeout)

**Синтаксис:**

```
err = report:ExportToFile(format, path, useTimeout)
```

**Результат:**

Сохраняет отчет `report` в формате `format` в файл по пути `path`. В случае успеха возвращает **nil**, иначе сообщение об ошибке. `useTimeout` - логическое значение, которое позволяет игнорировать таймауты времени исполнения серверного скрипта (если **false**) и используется для выгрузки отчетов с большим объемом данных.

Поддерживаемые форматы (значения переменной `format` в виде строки): `'csv'`, `'html'`, `'pdf'`, `'rtf'`, `'text'`, `'mht'`, `'xls'`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `format` | String | Формат отчета |
| `path` | String | Путь для сохранения отчета |
| `useTimeout` | Boolean | Логическое значение, которое позволяет игнорировать таймауты времени исполнения серверного скрипта и используется для выгрузки отчетов с большим объемом данных |

**Пример использования:** см. в описании метода *CreateDataSet()*.

## Print()

**Синтаксис:**

```
err = report:Print(printerName)
```

**Результат:**

Отправляет отчет `report` на принтер `printerName` для печати. В случае ошибки возвращает в `err` сообщение об ошибке, в случае успеха возвращает **nil**.

**Пример использования:** см. в описании метода *CreateDataSet()*.

# 14. Управление мнемосхемами

## 14.1. Методы

| Метод | Описание |
|---|---|
| *CreateDiagram (SymbolicName, DisplayName, Description)* | Позволяет создать мнемосхему. |
| *CreateDiagramFolder (SymbolicName, DisplayName, Description)* | Позволяет создать папку для мнемосхем. |
| *Delete()* | Позволяет удалить мнемосхему. |

## CreateDiagram()

**Синтаксис:**

```
local diagram, err = folder:CreateDiagram(symbolicName, displayName,
description)
```

**Результат:**

Создает мнемосхему.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `symbolicName` | String | Символьное имя новой мнемосхемы. |
| `displayName` | String | Отображаемое имя новой мнемосхемы, может быть пустой строкой. |
| `description` | String | Описание новой мнемосхемы, может быть пустой строкой. |

**Пример использования:**

```
local folder, err = Context:GetFolder('/Diagrams')
  if err~= nil then print(err) end
  local diagram, err = folder:CreateDiagram('DiagramSymbName', 'DiagramDispName',
  'DiagramDesc')
    if err ~= nill then print(err) end
```

В примере получаем папку - корневой каталог `Diagrams` и в полученной папке создаем мнемосхему `DiagramSymbName`.

## CreateDiagramFolder()

**Синтаксис:**

```
local subFolder, err = folder:CreateDiagramFolder(symbolicName, displayName,
description)
```

**Результат:**

Создает папку для мнемосхем.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `symbolicName` | String | Символьное имя новой папки. |
| `displayName` | String | Отображаемое имя новой папки, может быть пустой строкой. |
| `description` | String | Описание новой папки, может быть пустой строкой. |

**Пример использования:**

```
local folder, err = Context:GetFolder('/Diagrams')
if err ~= nil then print(err) end
  if folder ~= nil then
  local subFolder, err = folder:CreateDiagramFolder('SymbName','DispName',' ')
    if err ~= nil then print(er) end
end
```

В примере получаем папку - корневой каталог `Diagrams` и в полученной папке создаем папку `SymbName`.

## Delete()

**Синтаксис:**

```
err = source:Delete()
```

**Результат:**

Удаляет `source`, где `source` - мнемосхема. В случае успеха возвращает **nil**, иначе сообщение об ошибке.

**Пример использования:**

```
local diagr = Context:GetNode('/Diagrams/t1')

diagr:Delete()
```

В примере получаем мнемосхему `t1` из корневой папаки `Diagrams`, затем удаляем ее.

# 15. Работа с MODBUS протоколом

**Методы:**

| Имя | Описание |
|---|---|
| *modbus.open(host, port, receiveTimeout, writeTimeout)* | Создает новое modbus-подключение по заданному адресу |
| *connection:close()* | Закрывает подключение, созданное методом modbus.open() |
| *readCoils(slaveId, address, length)* | Возвращает заданное количество битов по указанному адресу, начиная с указанного регистра |
| *readDiscreteInputs(slaveId, address, length)* | Возвращает заданное количество дискретов по указанному адресу, начиная с указанного регистра |
| *readDoubleValues(slaveId, address, length, format)* | Возвращает заданное количество значений типа double по указанному адресу, начиная с указанного регистра по заданному порядку байтов |
| *readFloatValues(slaveId, address, length, format)* | Возвращает заданное количество значений типа float по указанному адресу, начиная с указанного регистра по заданному порядку байтов |
| *readHoldingRegisters(slaveId, address, length)* | Возвращает заданное количество значений по указанному адресу, начиная с указанного регистра |
| *readInputRegisters(slaveId, address, length)* | Возвращает заданное количество значений типа Input Registers по указанному адресу, начиная с указанного регистра |
| *readInt32Values(slaveId, address, length, format)* | Возвращает заданное количество значений типа Int32 по указанному адресу, начиная с указанного регистра по заданному порядку байтов |

| Имя | Описание |
|---|---|
| writeDoubleValues(slaveId, address, format, value1, value2, ...) | Записывает значения типа double по заданному адресу, в заданном формате, с указанного регистра |
| writeFloatValues(slaveId, address, format, value1, value2, ...) | Записывает значения типа float по заданному адресу, в заданном формате, с указанного регистра |
| writeInt32Values(slaveId, address, format, value1, value2, ...) | Записывает значения типа Int32 по заданному адресу, в заданном формате, с указанного регистра |
| writeMultipleCoils(slaveId, address, value1, value2, ...) | Записывает биты по заданному адресу, начиная с указанного регистра |
| writeMultipleRegisters(slaveId, address, value1, value2, ...) | Записывает значения по заданному адресу, начиная с указанного регистра |
| writeSingleCoil(slaveId, address, value) | Записывает бит по заданному адресу, в указанный регистр |
| writeSingleRegister(slaveId, address, value) | Записывает значение по заданному адресу, в указанный регистр |

## modbus.open()

**Синтаксис:**

```
connection, err = modbus.open('host','port',receiveTimeout,sendTimeout,retries)
```

**Результат:**

Создает новое modbus-подключение `connection` по адресу `host` с портом `port`, временем опроса `receiveTimeout`, временем записи `sendTimeout` и количеством попыток опроса устройства `retries` (не считая первой неудачной попытки). В случае ошибки возвращает в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `host` | String | Адрес modbus-подключения |
| `host` | Number | Порт modbus-подключения |
| `receiveTimeout` | Number | Время опроса |
| `sendTimeout` | Number | Время записи |
| `retries` | Number | Количество попыток опроса устройства, в случае неудачи |

**Пример использования:**

```
local connection = modbus.open('192.168.0.100', 502, 1000, 1000, 4)
local tag = Context:GetNode('/Tags/tag1')
local data = connection:readHoldingRegisters(1, 0, 1)
tag:SetValue(data[0])
connection:close()
```

В примере открываем соединение по адресу 192.168.0.100, получаем тег с именем `tag1`, читаем одно значение типа `Holding Register` с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

**ВАЖНО!**

По умолчанию, после первой неудачной попытки опроса устройства, будет предпринято еще 3 попытки. В этом случае, `retries` можно не указывать.

Если необходимо провести большее количество попыток опроса устройства, то следует указать их в `retries`.

## connection:close()

**Синтаксис:**

```
connection:close()
```

**Результат:**

Закрывает соединение `connection`, созданное функцией `modbus.open`.

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then return end
if connection ~= nil then
   err = connection:writeMultipleCoils(1, 0, 1, 1, 1, 1)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение с modbus-устройством и записываем биты в таблицу `Coils`, после чего соединение закрываем вне зависимости от успеха или ошибки записи.

## readCoils()

**Синтаксис:**

```
data, error = connection:readCoils(slaveId, address, length)
```

**Результат:**

Возвращает в `data` значения `length` битов modbus-устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля).

`connection` - это подключение, созданное функцией `modbus.open`.

В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства\| |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readCoils(1, 0, 1)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
   end
```

```
end

connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем один бит с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readDiscreteInputs()

**Синтаксис:**

```
data, error = connection:readDiscreteInputs(slaveId, address, length)
```

**Результат:**

Возвращает в `data` значения `length` дискретов modbus-устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства\| |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readDiscreteInputs(1, 0, 1)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
   end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно дискретное значение с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readDoubleValues()

**Синтаксис:**

```
data, error = connection:readDoubleValues(slaveId, address, length, format)
```

**Результат:**

В `data` возвращает `length` значений типа `double` modbus-устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате `format` (для прямого порядка байтов формат 012345678). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readDoubleValues(1, 0, 1, 12345678)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
```

```
    end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно значение типа `double` с прямым порядком байтов с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readFloatValues()

**Синтаксис:**

```
data, error = connection:readFloatValues(slaveId, address, length, format)
```

**Результат:**

В `data` возвращает `length` значений типа `float` modbus-устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате `format` (для прямого порядка байтов формат 01234). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в error сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readFloatValues(1, 0, 1, 1234)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
```

```
    end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно значение типа `float` с прямым порядком байтов с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readHoldingRegisters()

**Синтаксис:**

```
data, error = connection:readHoldingRegisters(slaveId, address, length)
```

**Результат:**

Возвращает в `data` значения типа `Holding Registers` в количестве `length`, modbus - устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля). `connection` - подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readHoldingRegisters(1, 0, 1)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
   end
```

```
end

connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно значение типа `Holding Register` с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readInputRegisters()

**Синтаксис:**

```
data, error = connection:readInputRegisters(slaveId, address, length)
```

**Результат:**

Возвращает в `data` значения типа `Input Registers` в количестве `length`, `modbus` - устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readInputRegisters(1, 0, 1)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
   end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно значение типа `Input Register` с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## readInt32Values()

**Синтаксис:**

```
data, error = connection:readInt32Values(slaveId,address,length,format)
```

**Результат:**

Возвращает в `data` значения типа `Int32` в количестве `length`, modbus-устройства с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате `format` (для прямого порядка байтов формат 01234). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в error сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `length` | Number | Длина битов modbus-устройства |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |

**Пример использования:**

```
local connection, err = modbus.open( '192.168.0.100 ', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   local data, err = connection:readInt32Values(1, 0, 1, 1234)
   if err ~= nil then
      print(err)
   else
      err = tag:SetValue(data[0])
      if err ~= ' ' then print(err) end
```

```
    end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, читаем одно значение типа `Int32` с прямым порядком байтов с нулевого регистра с модбас-устройства с идентификатором 1 и записываем полученное значение в тег.

## writeDoubleValues()

**Синтаксис:**

```
error = connection:writeDoubleValues(slaveId,address,format,value1,value2, ...)
```

**Результат:**

Записывает значения `value1`, `value2`, ..., типа `double` в modbus-устройство с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате `format` (для прямого порядка байтов формат 012345678). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |
| `value1,...` | Number | Значения типа double |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   if tag.Value == nil then connection:close() return end

   err = connection:writeDoubleValues(1, 0, 12345678, tag.Value)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, если значение тега не равно **nil**, то записываем его с прямым порядком байтов в нулевой регистр модбас-устройства с идентификатором 1 и закрываем соединение.

## writeFloatValues()

**Синтаксис:**

```
error = connection:writeFloatValues(slaveId,address,format,value1,value2, ...)
```

**Результат:**

Записывает значения `value1`, `value2`, ..., типа `float` в modbus-устройство с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате format (для прямого порядка байтов формат 01234). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |
| `value1,...` | Number | Значения типа float |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   if tag.Value == nil then connection:close() return end

   err = connection:writeFloatValues(1, 0, 1234, tag.Value)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, если значение тега не равно **nil**, то записываем его с прямым порядком байтов в нулевой регистр модбас-устройства с идентификатором 1 и закрываем соединение.

## writeInt32Values()

**Синтаксис:**

```
error = connection:writeInt32Values(slaveId,address,format,value1,value2, ...)
```

**Результат:**

Записывает значения `value1`, `value2`, ..., типа `Int32` в modbus-устройство с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля), в формате `format` (для прямого порядка байтов формат 01234). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `format` | Number | Формат (для прямого порядка байтов формат 012345678) |
| `value1,...` | Number | Значения типа Int32 |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then print(err) return end
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then
   print(err)
   connection:close()
   return
end
if connection ~= nil then
   if tag.Value == nil then connection:close() return end

   err = connection:writeInt32Values(1, 0, 1234, tag.Value)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение по адресу `192.168.0.100`, получаем тег с именем `tag1`, если значение тега не равно **nil**, то записываем его с прямым порядком байтов в нулевой регистр модбас-устройства с идентификатором 1 и закрываем соединение.

## writeMultipleCoils()

**Синтаксис:**

```
error = connection:writeMultipleCoils(slaveId,address,value1,value2, ...)
```

**Результат:**

Записывает биты `value1`, `value2`, ..., в modbus-устройство с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `value1,...` | Number | Значения |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then return end
if connection ~= nil then
   err = connection:writeMultipleCoils(1, 0, 1, 1, 1, 1)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение с modbus-устройством и записываем четыре бита в таблицу `Coils`, после чего соединение закрываем вне зависимости от успеха или ошибки записи.

## writeMultipleRegisters()

**Синтаксис:**

```
error = connection:writeMultipleRegisters(slaveId,address,value1,value2, ...)
```

**Результат:**

Записывает значения `value1`, `value2`, ..., в modbus-устройство с идентификатором `slaveId`, начиная с регистра `address` (нумерация регистров начинается с нуля). `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `value1,...` | Number | Значения |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then return end
if connection ~= nil then
   err = connection:writeMultipleRegisters(1, 0, 10, 20, 30, 40)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение с modbus-устройством и записываем в первые четыре регистра (начиная с нуля) значения 10, 20, 30, 40 в таблицу `Holding Registers`, после чего соединение закрываем вне зависимости от успеха или ошибки записи.

## writeSingleCoil()

**Синтаксис:**

```
error = connection:writeSingleCoil(slaveId, address, value)
```

**Результат:**

Записывает бит `value` в регистр `address` (нумерация регистров начинается с нуля) modbus - устройства с идентификатором `slaveId`. `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `value` | Number | Значение |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then return end
if connection ~= nil then
   err = connection:writeSingleCoil(1, 0, 1)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение с modbus-устройством и записываем бит в таблицу `Coils` в нулевой регистр устройства с адресом 1, после чего соединение закрываем вне зависимости от успеха или ошибки записи.

## writeSingleRegister()

**Синтаксис:**

```
error = connection:writeSingleRegister(slaveId, address, value)
```

**Результат:**

Записывает значение `value` в регистр `address` (нумерация регистров начинается с нуля) modbus-устройства с идентификатором `slaveId`. `connection` - это подключение, созданное функцией `modbus.open`. В случае ошибки возвращает в `error` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `slaveId` | Number | Идентификатор |
| `address` | Number | Регистр (нумерация регистров начинается с нуля) |
| `value` | Number | Значение |

**Пример использования:**

```
local connection, err = modbus.open('192.168.0.100', 502, 1500, 1500)
if err ~= nil then return end
if connection ~= nil then
   err = connection:writeSingleRegister(1, 0, 555)
   if err ~= nil then print(err) end
end
connection:close()
```

В примере открываем соединение с modbus-устройством и записываем `555` в таблицу `Holding Registers`, в нулевой регистр устройства с адресом 1, после чего соединение закрываем вне зависимости от успеха или ошибки записи.

# 16. Работа в веб-сервисами

Методы для работы с веб-сервисами доступны из глобальной таблицы webclient.

**Методы:**

| Имя | Описание |
|---|---|
| *webclient.downloadData(url)* | Возвращает массив байтов, полученный с указанного адреса |
| *webclient.downloadFile(dataUrl,filePath)* | Сохраняет полученное от веб-сервиса содержимое в файл по заданному пути |
| *webclient.downloadString(dataUrl)* | Возвращает данные от веб-сервиса в виде строки |

## webclient.downloadData()

**Синтаксис:**

```
bytes, err = webclient.downloadData(url)
```

**Результат:**

Возвращает массив байтов в `bytes`, в случае ошибки возвращает в `err` сообщение об ошибке и **nil** в `bytes`.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `url` | String | Адрес веб-сервиса |

**Пример использования:**

```
local bytes, err = webclient.downloadData('https://www.ya.ru/robots.txt')
if err ~= nil then print(err) return end
for i = 0,#bytes do
   print(bytes[i])
end
```

В примере мы получаем массив байтов в виде таблицы Lua и перебираем все элементы в цикле.

## webclient.downloadFile()

**Синтаксис:**

```
err = webclient.downloadFile(url, filePath)
```

**Результат:**

Сохраняет содержимое в файл по указанному пути `filePath`, в случае ошибки возвращает в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `url` | String | Адрес веб-сервиса |
| `filePath` | String | Путь для сохранения файла |

**Пример использования:**

```
local err = webclient.downloadFile('https://www.ya.ru/robots.txt', 'C:/Temp/robots.txt')
if err ~= nil then print(err) end
```

В примере получаем содержимое файла `robot.txt` и сохраняем в файл с таким же именем в папке `Temp` (файл создается автоматически, если файл с таким именем уже существует, то содержимое будет перезаписано).

## webclient.downloadString()

**Синтаксис:**

```
result, err = webclient.downloadString(url)
```

**Результат:**

Возвращает в `result` данные в виде строки, в случае ошибки возвращает в `err` сообщение об ошибке.

**Аргументы метода:**

| Имя | Тип | Описание |
|---|---|---|
| `url` | String | Адрес веб-сервиса |

**Пример использования:**

```
local result, err = webclient.downloadString('https://www.ya.ru/robots.txt')
if err ~= nil then
   print(result)
else
   print(err)
end
```

В примере получаем содержимое файла `robot.txt` в переменную `result` и выводим содержимое на печать.

# 17. Реляционный доступ к серверным объектам

## Таблица пользователей - users

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| Id | INTEGER | Int64 | number | Идентификатор пользователя |
| SymbolicName | TEXT | String | string | Символьное наименование |
| DisplayName | TEXT | String | string | Отображаемое наименование |
| Description | TEXT | String | string | Описание |

**Пример использования:**

```
local q, err = Context:Query('SELECT * FROM users')
if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end

 if c ~= true then break; end

 local count, err = r:GetFieldCount()
 if err ~= nil then print('GetFieldCount: ' .. err); return; end

 for i = 0,count-1 do
  local fn, err = r:GetFieldName(i)
  if err ~= nil then print('GetFieldName: ' .. err); return; end

  local ft, err = r:GetFieldType(i)
  if err ~= nil then print('GetFieldType: ' .. err); return; end

  local value;
  local isNull, err = r:IsDbNull(i)
  if err ~= nil then print('IsDbNull: ' .. err); return; end
```

```
  if isNull then
   value = '<nil>'
  elseif ft == 'INTEGER' then
   value, err = r:GetInt(i)
   if err ~= nil then print('GetInt: ' .. err); return; end
  elseif ft == 'TEXT' then
   value, err = r:GetString(i)
   if err ~= nil then print('GetString: ' .. err); return; end
  else
   value = '<Unknown type: "' .. ft .. '">'
  end
  print(fn .. '[' .. ft .. ']' .. ' ' .. value)
 end

 print()
end

r:Close()
q:Close()
```

## Таблица ролей - roles

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| Id | INTEGER | Int64 | number | Идентификатор роли |
| SymbolicName | TEXT | String | string | Символьное наименование |
| DisplayName | TEXT | String | string | Отображаемое наименование |
| Description | TEXT | String | string | Описание |

**Пример использования:**

```
local q, err = Context:Query('SELECT * FROM roles')
if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end

 if c ~= true then break; end

 local count, err = r:GetFieldCount()
 if err ~= nil then print('GetFieldCount: ' .. err); return; end

 for i = 0,count-1 do
  local fn, err = r:GetFieldName(i)
  if err ~= nil then print('GetFieldName: ' .. err); return; end

  local ft, err = r:GetFieldType(i)
  if err ~= nil then print('GetFieldType: ' .. err); return; end

  local value;
  local isNull, err = r:IsDbNull(i)
  if err ~= nil then print('IsDbNull: ' .. err); return; end
```

```
  if isNull then
   value = '<nil>'
  elseif ft == 'INTEGER' then
   value, err = r:GetInt(i)
   if err ~= nil then print('GetInt: ' .. err); return; end
  elseif ft == 'TEXT' then
   value, err = r:GetString(i)
   if err ~= nil then print('GetString: ' .. err); return; end
  else
   value = '<Unknown type: "' .. ft .. '">'
  end
  print(fn .. '[' .. ft .. ']' .. ' ' .. value)
 end

 print()
end

r:Close()
q:Close()
```

## Таблица связи пользователей и ролей - users_roles

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| UserId | INTEGER | Int64 | number | Идентификатор пользователя |
| RoleId | TEXT | String | string | Идентификатор роли |

**Пример использования:**

```
local q, err = Context:Query([[
SELECT users.SymbolicName, roles.SymbolicName, users.Id
  FROM users_roles
  JOIN users ON users_roles.UserId = users.Id
  JOIN roles ON users_roles.RoleId = roles.Id
  ]])
if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end

 if c ~= true then break; end

 local u,err = r:GetString(0)
 if err ~= nil then print('GetString: ' .. err); return; end
 local g,err = r:GetString(1)
 if err ~= nil then print('GetString: ' .. err); return; end
 local uid,err = r:GetInt(2)
 if err ~= nil then print('GetInt: ' .. err); return; end

 if uid == Context.CurrentUserId then u = '<' .. u .. '>'; end

 print(u..'\t'..g)
end
```

```
r:Close()

q:Close()
```

## Таблица тегов - tags

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| Id | INTEGER | Int64 | number | Идентификатор тега |
| ParentId | INTEGER | Int64 | number | Идентификатор родительского объекта |
| SymbolicName | TEXT | String | string | Символьное наименование |
| DisplayName | TEXT | String | string | Отображаемое наименование |
| Description | TEXT | String | string | Описание |
| IntValue | INTEGER | Int64 | number | Целочисловое значение тега |
| DoubleValue | REAL | Double | number | Вещественное значение тега |
| StringValue | TEXT | String | string | Строковое значение тега |
| BytesValue | BLOB | Byte[] | object | Бинарное значение тега |
| StatusCode | INTEGER | Int64 | number | Код состояние |
| Timestamp | INTEGER | Int64 | number | Отметка времени значения в тиках |
| BindingId | INTEGER | Int64 | number | Идентификатор биндинга |
| TargetId | INTEGER | Int64 | number | Идентификатор целевого тега |
| Expression | TEXT | String | string | Выражения вычисляемого тега |

**Пример использования:**

```
local q, err = Context:Query('SELECT * FROM tags LIMIT 10')
if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end

 if c ~= true then break; end
```

```
local count, err = r:GetFieldCount()
if err ~= nil then print('GetFieldCount: ' .. err); return; end

for i = 0,count-1 do
 local fn, err = r:GetFieldName(i)
 if err ~= nil then print('GetFieldName: ' .. err); return; end

 local ft, err = r:GetFieldType(i)
 if err ~= nil then print('GetFieldType: ' .. err); return; end

 local value;
 local isNull, err = r:IsDbNull(i)
 if err ~= nil then print('IsDbNull: ' .. err); return; end

 if isNull then
  value = '<nil>'
 elseif ft == 'INTEGER' then
  if fn == 'Timestamp' then
   local dt, err = r:GetDateTime(i)
   if err ~= nil then print('GetDateTime: ' .. err); return; end
   value = dt:ToString("yyyy-MM-dd HH:mm:ss.fffff")
  else
   value, err = r:GetInt(i)
   if err ~= nil then print('GetLong: ' .. err); return; end
  end
 elseif ft == 'TEXT' then
  value, err = r:GetString(i)
  if err ~= nil then print('GetString: ' .. err); return; end
 elseif ft == 'REAL' then
  value, err = r:GetDouble(i)
  if err ~= nil then print('GetDouble: ' .. err); return; end
 elseif ft == 'BLOB' then
  value, err = r:GetBytes(i)
  if err ~= nil then print('GetBytes: ' .. err); return; end
  value = '<' .. value.Length .. ' Byte(s)>'
 else
```

```
    value = '<Unknown type: "' .. ft .. '">'
   end
   print(fn .. '[' .. ft .. ']' .. ' ' .. value)
  end


  print()
end


r:Close()
q:Close()
```

***Прим.:***

- *Конвертирования значения тега не происходит, то есть, если тег имеет тип Int, то в колонках DoubleValue, StringValue и BytesValue будет NULL;*
- *Если к тегу не привязан биндинг, то в колонке BindingId будет NULL;*
- *Если тег не является ссылкой, то в колонке TargetId будет NULL.*

## Таблица папок - folders

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| Id | INTEGER | Int64 | number | Идентификатор папки |
| ParentId | INTEGER | Int64 | number | Идентификатор родительской папки |
| SymbolicName | TEXT | String | string | Символьное наименование |
| DisplayName | TEXT | String | string | Отображаемое наименование |
| Description | TEXT | String | string | Описание |

**Пример использования:**

```
local q, err = Context:Query([[
WITH RECURSIVE
  tree(id, level, SymbolicName) AS (
    VALUES(NULL, 0, 'Root')
    UNION ALL
    SELECT folders.Id, tree.level + 1, folders.SymbolicName
      FROM folders
      JOIN tree
      ON coalesce(folders.ParentId, 0) = coalesce(tree.Id, 0)
     ORDER BY 2 DESC
  )
SELECT substr('.............', 1, level) || SymbolicName FROM tree;
]])

if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end

 if c ~= true then break; end
```

```
 local value, err = r:GetString(0)
 print(value)
end


r:Close()
q:Close()
```

***Прим.:***

- *Если папка корневая (Templates, Tags, Agents, Diagrams, Jobs), то в колонке ParentId будет NULL;*
- *Папки Users и Roles не представлены в таблице folders.*

## Таблица исторических значений - hvalues

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| TagId | INTEGER | Int64 | number | Идентификатор тега-источника значения |
| SourceTimestamp | INTEGER | Int64 | number | Отметка времени значения в тиках |
| ServerTimestamp | INTEGER | Int64 | number | Отметка времени сохранения значения в тиках |
| StatusCode | INTEGER | Int64 | number | Код состояние |
| IntValue | INTEGER | Int64 | number | Целочисловое значение тега |
| DoubleValue | REAL | Double | number | Вещественное значение тега |
| StringValue | TEXT | String | string | Строковое значение тега |
| BytesValue | BLOB | Byte[] | object | Бинарное значение тега |

**Пример использования:**

```
local tag = Context:GetNode('/Tags/d1')
local tagId = tag.Id
local bottomBound = 536773342871968899
local topBound = 736773342871968899
local sql = string.format('SELECT COUNT(*) as Count, AVG(DoubleValue) as Avg FROM
 hvalues WHERE TagId = %d AND SourceTimestamp BETWEEN %.0f AND %.0f;', tagId, topBound,
 bottomBound)
print(sql)
local q, err = Context:Query(sql)
if err ~= nil then print('Query: ' .. err); return; end

local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end

while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end
```

```
if c ~= true then break; end


local count, err = r:GetFieldCount()
if err ~= nil then print('GetFieldCount: ' .. err); return; end


for i = 0,count-1 do
 local fn, err = r:GetFieldName(i)
 if err ~= nil then print('GetFieldName: ' .. err); return; end


 local ft, err = r:GetFieldType(i)
 if err ~= nil then print('GetFieldType: ' .. err); return; end


 local value;
 local isNull, err = r:IsDbNull(i)
 if err ~= nil then print('IsDbNull: ' .. err); return; end


 if isNull then
  value = '<nil>'
 elseif ft == 'INTEGER' then
  if fn == 'SourceTimestamp' or fn == 'ServerTimestamp' then
   local dt, err = r:GetDateTime(i)
   if err ~= nil then print('GetDateTime: ' .. err); return; end
   value = dt:ToString("yyyy-MM-dd HH:mm:ss.fffff")
  else
   value, err = r:GetInt(i)
   if err ~= nil then print('GetInt: ' .. err); return; end
  end
 elseif ft == 'TEXT' then
  value, err = r:GetString(i)
  if err ~= nil then print('GetString: ' .. err); return; end
 elseif ft == 'REAL' then
  value, err = r:GetDouble(i)
  if err ~= nil then print('GetDouble: ' .. err); return; end
 elseif ft == 'BLOB' then
  value, err = r:GetBytes(i)
  if err ~= nil then print('GetBytes: ' .. err); return; end
  value = '<' .. value.Length .. ' Byte(s)>'
```

```
  else
   value = '<Unknown type: "' .. ft .. '">'
  end
  print(fn .. '[' .. ft .. ']' .. ' ' .. value)
 end


 print()
end


r:Close()
q:Close()
```

***Прим.:***

- *Конвертирования значения тега не происходит, то есть, если тег имеет тип Int, то в колонках DoubleValue, StringValue и BytesValue будет NULL;*
- *Запросы без ограничений значения полей TagId и SourceTimestamp будут возвращать ошибку: No valid constraint present for TagId and SourceTimestamp.*

## Таблица событий

| Колонка | Тип SQL | Тип .NET | Тип Lua | Описание |
|---|---|---|---|---|
| EventId | INTEGER | Int64 | number | Идентификатор события |
| SourceId | INTEGER | Int64 | number | Идентификатор источника события |
| Timestamp | INTEGER | Int64 | number | Отметка времени сохранения значения в тиках |
| ReceiveTimestamp | INTEGER | Int64 | number | Отметка времени получения значения в тиках |
| SourceName | TEXT | String | string | Наименование источника события |
| Severity | INTEGER | Int64 | number | Важность события |
| Message | TEXT | String | string | Сообщение события |
| AckedTimestamp | INTEGER | Int64 | number | Отметка времени квитирования события в тиках |
| AckedWho | TEXT | String | string | Наименование пользователя квитировавшего событие |
| AckedWhoText | TEXT | String | string | Наименование пользователя квитировавшего событие |
| AckedWhere | TEXT | String | string | Наименование приложения квитировавшего событие |

**Пример использования:**

```
local folder = Context:GetFolder('/Tags/Heating')
local folderId = folder.Id
local bottomBound = 536773342871968899
local topBound = 736773342871968899
local sql = string.format('SELECT * FROM events WHERE SourceId = %d AND Timestamp BETWEEN
 %.0f AND %.0f LIMIT 10;', folderId, bottomBound, topBound)
print(sql)
local q, err = Context:Query(sql)
```

```
if err ~= nil then print('Query: ' .. err); return; end


local r, err = q:ExecuteReader()
if err ~= nil then print('ExecuteReader: ' .. err); return; end


while true do
 local c,err = r:Read()
 if err ~= nil then print('Read: ' .. err); return; end


 if c ~= true then break; end


 local count, err = r:GetFieldCount()
 if err ~= nil then print('GetFieldCount: ' .. err); return; end


 for i = 0,count-1 do
  local fn, err = r:GetFieldName(i)
  if err ~= nil then print('GetFieldName: ' .. err); return; end


  local ft, err = r:GetFieldType(i)
  if err ~= nil then print('GetFieldType: ' .. err); return; end


  local value;
  local isNull, err = r:IsDbNull(i)
  if err ~= nil then print('IsDbNull: ' .. err); return; end


  if isNull then
   value = '<nil>'
  elseif ft == 'INTEGER' then
   if fn == 'Timestamp' or fn == 'ReceiveTimestamp' or fn == 'AckedTimestamp' then
    local dt, err = r:GetDateTimeUtc(i)
    if err ~= nil then print('GetDateTime: ' .. err); return; end
    value = dt:ToString("yyyy-MM-dd HH:mm:ss.fffff")
   else
    value, err = r:GetInt(i)
    if err ~= nil then print('GetInt: ' .. err); return; end
   end
  elseif ft == 'TEXT' then
```

```
    value, err = r:GetString(i)
    if err ~= nil then print('GetString: ' .. err); return; end
   elseif ft == 'REAL' then
    value, err = r:GetDouble(i)
    if err ~= nil then print('GetDouble: ' .. err); return; end
   else
    value = '<Unknown type: "' .. ft .. '">'
   end
   print(fn .. '[' .. ft .. ']' .. ' ' .. value)
  end

  print()
end

r:Close()
q:Close()
```

***Прим.:***

- *Если событие не квитировано, то в колонках AckedTimestamp, AckedWho, AckedWhoText и AckedWhere будет NULL;*
- *Запросы без ограничения SourceId и Timestamp или только Timestamp будут возвращать ошибку: No valid constraint present for Timestamp and SourceId or Timestamp.*

# 18. Дополнительные возможности

## 18.1. Свойства среды исполнения программы `Context:`

| Имя | Тип | Описание |
|---|---|---|
| *CurrentUserDisplayName* | String | Возвращает отображаемое имя пользователя, который инициировал запуск программы |
| *StopPending* | Boolean | Возвращает и устанавливает флаг запроса останова программы |
| *ChangedTag* | ChangedTagInfo | Возвращает объект с текущим и предыдущим значением тега (доступно при запуске программы по изменению тега или шаблона) |

## CurrentUserDisplayName

**Синтаксис:**

```
local userName = Context.CurrentUserDisplayName
```

**Результат:**

Возвращает отображаемое имя пользователя, который инициировал запуск программы (job).

**Пример использования:**

```
local source = Context:GetFolder('/Tags')
if source ~= nil then
local event, err = source:ReportEvent('test',300,
Context.CurrentUserDisplayName..' запустил программу.')
if err ~= nil then print(err) end
end
```

В примере генерируем событие, в котором выводим сообщение с именем пользователя + `' запустил программу'`.

## StopPending

**Синтаксис:**

```
bool = Context.StopPending
```

**Результат:**

Возвращает флаг запроса останова программы (job).

**Пример использования:**

```
local tag, err = Context:GetNode('/Tags/tag1')
if err ~= nil then print(err) return end
while not Context.StopPending do
  err = tag:SetValue(Context.Random:Next())
  if err ~= nil then print(err) end
  os.sleep(1000)
end
```

В примере пишем случайные значения в тег `'tag1'` до тех пор, пока не будет выставлен флаг запроса останова программы.

## ChangedTag

**Синтаксис:**

```
local tagInfo = Context.ChangedTag
```

**Результат:**

Возвращает объект с новым и предыдущим значением тега (доступно при запуске программы по изменению тега или шаблона).

**Доступные свойства:**

| Имя | Тип | Описание |
|---|---|---|
| `Path` | String | Полный путь к тегу, инициировавшему запуск программы |
| `OldValue` | Object | Предыдущее значение тега |
| `NewValue` | Object | Новое значение тега |
| `OldStatusCode` | Number | Код статуса предыдущего значения |
| `NewStatusCode` | Number | Код статуса нового значения |
| `OldTimestamp` | DateTime | Отметка времени предыдущего значения |
| `NewTimestamp` | DateTime | Отметка времени нового значения |

**Пример использования:**

```
local o = Context.ChangedTag
print(o.Path)
print(o.OldValue)
print(o.NewValue)
print(o.OldStatusCode)
print(o.NewStatusCode)
print(o.OldTimestamp)
print(o.NewTimestamp)
```

При запуске программы по изменению тега на печать будут выведены все доступные свойства объекта.

# 18.2. Дополнительные методы

| Имя | Описание |
|---|---|
| *BackupDatabase (name)* | Создает полную резервную копию БД |
| *GetPath ()* | Возвращает полный путь к программе, из которой вызвана функция |
| *GetPrinterNames ()* | Возвращает коллекцию установленных принтеров в виде массива строк |
| *IsCurrentUserInRole (roleSymbolicName)* | Проверяет наличие пользователя в указанной роли и возвращает результат в виде логического значения |
| *os.sleep (pause)* | Приостанавливает выполнение программы на заданное количество миллисекунд |
| *Require (jobPath)* | Подключает программу по указанному пути в качестве библиотеки |
| *ResetWatchdogTimer ()* | Сбрасывает таймер выполнения программы, запущенной по расписанию или изменению тега (шаблона тега) |
| *TableToByteArray (table)* | Возвращает массив байтов, конвертированный из указанной таблицы Lua |
| *Random:Next ()* | Возвращает случайное целое неотрицательное число |
| *Random:Next (number)* | Возвращает случайное целое неотрицательное число, которое будет меньше заданного number |
| *Random:Next (low, high)* | Возвращает случайное целое число, находящееся в промежутке между заданными значениями lowNumber, highNumber |
| *Random:NextDouble ()* | Возвращает случайное вещественное число, находящееся в промежутке между [0.0 и 1.0) |

| Имя | Описание |
|---|---|
| GetAllEndPoint () | Позволяет получить все EndPoint сервера в формате строки |
| GetServerName () | Позволяет получить имя машины, на которой работает текущий сервер |
| GetBytesFromChar () | Возвращает набор байтов из символьной переменной |
| GetBytesFromBool () | Возвращает набор байтов из логической переменной |
| GetBytesFromInt16 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [-32768 : 32767] |
| GetBytesFromInt32 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [-2147483648 : 2147483647] |
| GetBytesFromInt64 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [-9223372036854775808 : 9223372036854775807] |
| GetBytesFromUInt16 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [0 : 65535] |
| GetBytesFromUInt32 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [0 : 4294967295] |
| GetBytesFromUInt64 () | Возвращает набор байтов из целочисленной переменной в диапазоне значений [0 : 18446744073709551615] |
| GetBytesFromSingle () | Возвращает набор байтов из десятичной переменной с плавающей точкой |
| GetBytesFromDouble () | Возвращает набор байтов из десятичной переменной с плавающей точкой двойной точности |

**ВАЖНО!**

При работе с функциями GetBytesFromChar () - GetBytesFromDouble () обязательно начинайте массив LUA с 0, в целях сохранения порядка данных в массиве.

## BackupDatabase()

**Синтаксис:**

```
Context:BackupDatabase(name)
```

**Результат:**

Создает полную резервную копию БД в каталоге `C:/ProgramData/KSoft/backups` с именем `name`.

**Пример использования:**

```
local name = 'myBackup'

Context:BackupDatabase(name)
```

В примере создаем бэкап с именем `'myBackup'`.

## GetPath()

**Синтаксис:**

```
local path = Context:GetPath()
```

**Результат:**

Возвращает путь к программе (job), из которой вызвана функция.

**Пример использования:**

```
local path = Context:GetPath()
print(path)
```

В примере для программы с именем `test` из корневого каталога Jobs функция вернет строку `'/Jobs/test'`.

## GetPrinterNames()

**Синтаксис:**

```
printers, err = Context:GetPrinterNames()
```

До версии 3.3.19:

```
err, printers = Context:GetPrinterNames()
```

**Результат:**

Возвращает **nil** в `err` и коллекцию установленных принтеров в `printers` в виде массива строк, в случае ошибки возвращает сообщение об ошибке в `err`.

**Пример использования:**

```
local printers, err = Context:GetPrinterNames()
if err ~= nil then print(err) return end
for i = 0, printers.Length - 1 do
    print(printers[i])
end
```

В примере получаем массив строк с установленных именами принтеров и выводим их по одному на печать. В случае ошибки получения массива завершаем работу программы с помощью ключевого слова `return`. Функция используется для получения имени принтера при реализации автоматической печати отчетов.

## IsCurrentUserInRole()

**Синтаксис:**

```
bool = IsCurrentUserInRole(roleSymbolicName)
```

**Результат:**

Возвращает **true**, если пользователь, запустивший программу состоит в роли с символьным именем `roleSymbolicName`, иначе возвращает **false**.

**Пример использования:**

```
local tag = Context:GetNode('/Tags/tag1')
if tag ~= nil then
    if Context:IsCurrentUserInRole('admins') then
        local err = tag:SetValue(Context.Random:Next(0,10))
        if err ~= nil then print(err) end
    else
        local source = tag.Parent
               source:ReportEvent(source.SymbolicName, 700, 'Access denied for:' ..
 Context.CurrentUserDisplayName)
    end
end
```

В примере получаем тег `tag1` из корневого каталога `Tags`, затем проверяем наличие пользователя, запустившего программу в роли `admins`, если пользователь состоит в роли, то записываем в тег случайное значение от `1` до `10`, в противном случае генерируем событие об отсутствии доступа.

## os.sleep()

**Синтаксис:**

```
os.sleep(pause)
```

**Результат:**

Приостанавливает выполнение скрипта на `pause` миллисекунд.

**Пример использования:**

```
while not Context.StopPending do
    print(os.date("%d.%m.%Y %H:%M:%S"))
    os.sleep(1000)
end
```

В примере выводится на печать дата и время с периодичностью раз в секунду до тех пор, пока не будет выставлен флаг запроса останова программы.

## Require()

**Синтаксис:**

```
Context:Require(jobPath)
```

**Результат:**

Подключает серверную программу по пути `jobPath` в качестве библиотеки.

**Пример использования:**

```
Context:Require('/libs/utils')
```

В примере добавляем в качестве библиотеки программу с именем `utils`, полный путь к которой выглядит следующим образом: `/Jobs/libs/utils`.

## ResetWatchdogTimer()

**Синтаксис:**

```
Context:ResetWatchdogTimer()
```

**Результат:**

Сбрасывает таймер выполнения программы. Используется в программах, запускаемых по расписанию или изменению тега (шаблона тега). По умолчанию, такие программы принудительно завершаются через 10 секунд после запуска выполнения.

**Пример использования:**

```
Context:ResetWatchdogTimer()
```

## TableToByteArray()

**Синтаксис:**

```
data, err = Context:TableToByteArray(table)
```

**Результат:**

Возвращает массив байтов, конвертированный из таблицы байтов `table`, в `data` и **nil** в `err` в случае успеха, в случае ошибки возвращает **nil** в `data` и сообщение об ошибке в `err`.

**Пример использования:**

```
local tagByteArray = Context:GetTag('/tByteArray')
if tagByteArray == nil then return end
local table = {10, 100, 200}
local data, err = Context:TableToByteArray(table)
if err ~= nil then print(err) return end
err = tagByteArray:SetValue(data)
if err ~= '' then print(err) end</code>
```

В примере получаем тег типа `ByteArray`, конвертируем таблицу байтов `table` в массив байтов `data` и записываем полученный массив в тег.

## Random:Next()

**Результат:**

Возвращает случайное целое неотрицательное число.

**Пример использования:**

```
local t = Context:GetNode('/Tags/folder/tag')

t:SetValue(Context.Random:Next())

print(t.Value)
```

В примере присваиваем тегу значение и выводим на печать - случайное целое неотрицательное число.

## Random:Next(number)

**Результат:**

Возвращает случайное целое неотрицательное число, которое будет меньше заданного `number`.

**Пример использования:**

```
local t = Context:GetNode('/Tags/folder/tag')

t:SetValue(Context.Random:Next(20))

print(t.Value)
```

В примере присваиваем тегу значение и выводим на печать - случайное целое неотрицательное число, которое будет меньше `20`.

## Random:Next(low, high)

**Результат:**

Возвращает случайное число, находящееся в промежутке между заданными значениями `low`, `high`.

**Пример использования:**

```
-- пример для целого неотрицательного числа  в диапазоне от 20 до 70
local t = Context:GetNode('/Tags/folder/tag_int')
t:SetValue(Context.Random:Next(20,70))
print(t.Value)

-- пример для дробного неотрицательного числа  в диапазоне от 0,0 до 10,0
local t = Context:GetNode('/Tags/folder/tag_double')
t:SetValue(Context.Random:Next(0,1000)*0.01)
print(t.Value)
```

В примере присваиваем тегу значение и выводим на печать.

## Random:NextDouble()

**Результат:**

Возвращает случайное вещественное число, которое будет больше или равно `0.0` и меньше `1.0`.

**Пример использования:**

```
local t = Context:GetNode('/Tags/folder/tag')

t:SetValue(Context.Random:NextDouble())

print(t.Value)
```

В примере присваиваем тегу значение и выводим на печать - случайное вещественное число.

## GetAllEndPoint()

**Результат:**

Возвращает все `EndPoint` сервера в формате строки.

**Пример использования:**

```
local tag, err = Context:GetNode('/Project/Xtree/Ex2')
  while not Context.StopPending do
  local endpoint = Context:GetServerEndPoints()
    if tag.Value ~= endpoint
    then err = tag:SetValue(endpoint)
  end
os.sleep(5000)
end
```

В примере в локальную переменную `tag` получаем тег `Ex2`, в локальную переменную `endpoint` получаем все `EndPoint` сервера, затем значению переменной `tag` присваиваем значение переменной `endpoint`.

## GetServerName()

**Результат:**

Возвращает имя машины, на которой работает текущий сервер.

**Пример использования:**

```
local tag, err = Context:GetTag('/Project/Xtree/Ex2')
while not Context.StopPending do
  local name = Context:GetServerName()
  if tag.Value ~= name
  then err = tag:SetValue(name)
end
os.sleep(5000)
end
```

В примере в локальную переменную `tag` получаем тег `Ex2`, в локальную переменную `name` получаем имя сервера, затем значению переменной `tag` присваиваем значение переменной `name`.

## GetBytesFromInt16 ()

**Синтаксис:**

```
local byteArray = Context:GetBytesFromInt16(data)
```

**Результат:**

Возвращает набор байтов из целочисленной переменной в диапазоне значений [-32768 : 32767].

**Пояснение:**

Общий алгоритм для всех методов при работе с протоколом modbus:

1. Получить методом `GetBytesFromInt32()` байты из переменной (каждый метод относится к отдельному типу переменной).
2. Преобразовать полученный результат в нужный формат данных.

**Пример использования:**

```
local ipModbus = '127.0.0.1'
local portModbus = 502
local slaveId = 1
local address = 12
local length = 1
local bitIndex = 1
-- формат писать с 0 пример: 0123, 1032, 2301, 3210
format = {}
format[0] = 0
format[1] = 1
--------------------------------------------------------------------------------------------
local source = Context:GetTagFolder('/')
source:ReportEvent(source.SymbolicName, 123, 'Старт модбас Тест')
local connection, err = modbus.open(ipModbus, portModbus, 1500, 1500)
source:ReportEvent(source.SymbolicName, 123, 'Успешное подключение')
local data, err = connection:readHoldingRegisters(slaveId, address, length)
b1 = Context:GetBytesFromInt32(data[0])
rawBytes = {}
rawBytes[0] = b1[0]
rawBytes[1] = b1[1]
bytes = {}
for i = 0, #format do
```

```
bytes[i] = tonumber(rawBytes[format[i]])
end
local value = Context:ToWordBit(bytes, bitIndex)
source:ReportEvent(source.SymbolicName, 123, 'Значение: '..tostring(value))
connection:close()
```

***Прим.:*** *Для различных типов данных реализованы соответствующие методы: GetBytesFromChar (),GetBytesFromBool(), GetBytesFromInt16(), GetBytesFromInt32(), GetBytesFromInt64(), GetBytesFromUInt16(), GetBytesFromUInt32(), GetBytesFromUInt64(), GetBytesFromSingle(), GetBytesFromDouble().*

# 18.3. Методы, не рекомендуемые к использованию, начиная с KSE Platform версии 3.4.8

В данном разделе приведен список устаревших методов.

***Прим.:*** *Ранее созданные проекты сохраняют свою работоспособность, но после версии 3.4.8 использование устаревших методов не рекомендуется.*

| **Имя** | **Описание** |
|---|---|
| `DeleteWithAgents()` | Удаляет каталог и вложенные агенты |
| `DeleteWithTags()` | Удаляет каталог и вложенные теги |
| `DeleteWithDiagrams()` | Удаляет каталог и вложенные мнемосхемы |
| `DeleteWithTemplates()` | Удаляет каталог и вложенные шаблоны тегов |
| `DeleteWithJobs()` | Удаляет каталог и вложенные джобы (программы) |
| `GetAgent()` | Возвращает агента с указанным символьным именем из каталога |
| `GetAgentFolder()` | Возвращает каталог из корня Agents |
| `GetTag()` | Возвращает тег в папке |
| `GetTags()` | Возвращает коллекцию тегов, вложенных в папку |
| `GetTagFolder()` | Возвращает каталог из корня Tags |
| `GetTagTemplate()` | Возвращает имя шаблона тега |
| `GetTagTemplates()` | Возвращает коллекцию шаблонов тегов, вложенных в папку |
| `GetTagTemplateFolder()` | Возвращает каталог из корня Templates |
| `GetJob()` | Возвращает программу в переменную |

| **Имя** | **Описание** |
|---|---|
| `GetJobs()` | Возвращает коллекцию джобов, вложенных в папку |
| `GetJobFolder()` | Возвращает папку из корневого каталога Jobs |
| `GetDiagram()` | Возвращает имя мнемосхемы в переменную |
| `GetDiagrams()` | Возвращает коллекцию мнемосхем, вложенных в папку |
| `GetDiagramFolder()` | Возвращает папку из корневого каталога Diagrams |