PHOENIX
DYNAMIC SYSTEMS
ПРОГРАММНАЯ ПЛАТФОРМА

Руководство администратора

Программный модуль

PhoenixDS CmpEvnt

Аннотация

Программный модуль "PhoenixDS CmpEvnt" (далее – ПМ **PhoenixDS CmpEvnt**), расширяющий базовый функционал Программной платформы "PhoenixDS" (далее – ПП PhoenixDS), позволяет настраивать сценарии дополнительной обработки данных, полученных от различных источников, с применением нестандартной логики. Может использоваться для агрегирования данных, а также для обработки сторонних данных при интеграции систем.

В настоящем Руководстве администратора приведен порядок настройки объектов, посредством которых реализуется нестандартная обработка данных.

Перед изучением настоящего документа рекомендуется освоить основные принципы функционирования ПП PhoenixDS, изложенные в базовом документе "ПП PhoenixDS. Руководство администратора".

Россия, 101066, Москва, ул. Бауманская, д.53

тел.: +7 (495) 228-1100

www.dynasystems.ru

© ООО «Динамические Системы»

Документ является собственностью ООО «Динамические Системы» и защищен законодательством о правах на результаты интеллектуальной деятельности. Никакая информация, содержащаяся в данном документе, не может быть воспроизведена, искажена, переработана, переведена на иностранный язык, записана или скопирована для любых коммерческих целей. Не допускается передача данного документа третьей стороне без письменного согласия ООО «Динамические Системы». Графические изображения и названия продуктов, упоминаемые в данном документе, могут быть зарегистрированными товарными знаками, охраняемыми законодательством о правах на результаты интеллектуальной деятельности. ООО «Динамические Системы» оставляет за собой право вносить изменения в содержание данного документа без предварительного уведомления.

Типографские соглашения и условные обозначения

В документе приняты следующие шрифты и обозначения:

Примеры использования	Тип информации
Дерево конфигурации – это... кнопка **Сохранить** пункт меню **Конфигурировать**	Термины, названия компонент ПП PhoenixDS, наименования элементов пользовательского интерфейса (поля, кнопки, пункты меню).
... выбрать значение **off** ...	Значения полей, отображаемых в пользовательском интерфейсе.
`konfne --devlist` ... директория `C:\Program Files (x86)\` ...	Команды, фрагменты кода программы, текст файлов.
См. Справочники 49 ... с помощью дополнительной настройки 12 ...	Ссылка на другой раздел документа. Значок возле ссылки содержит номер страницы, на которой расположен целевой раздел.

В абзацах с заголовком Примечание содержатся дополнительные сведения, например:

> Примечание. Таким способом представлена дополнительная информация по теме раздела.

В ограниченные рамкой текстовые блоки с заголовком **ВНИМАНИЕ!** помещены требования или инструкции, невыполнение которых может привести к необратимым последствиям, а также особо важная информация, например:

> **ВНИМАНИЕ!** Таким способом выделена информация, которую нужно обязательно принять к сведению.

Содержание

1. Введение 5

1.1. Назначение ПМ PhoenixDS CmpEvnt 6

2. Настройка и использование ПМ PhoenixDS CmpEvnt 7

2.1. Устройство комплексной логики (Devices::Analysis::Logix) 8

2.2. Структура окна создания устройства 8

2.3. Создание устройства 9

2.4. Код для вычисления значения источника данных 10

2.5. Схема для вычисления значения источника данных 12

2.5.1. Представление Схема 12

2.5.1.1. Добавление функции/правила в сценарий 13

2.5.1.2. Настройка правила 14

2.5.2. Текстовое представление 16

2.5.3. Древовидное представление 17

2.6. Удаление сценария с устройства комплексной логики 17

2.7. Функции 18

2.7.1. Логические функции (AND, OR, ANY, NOT, NAND, NOR, XOR) 18

2.7.2. Временые функции (TON, TOF, STOF) 19

2.7.3. Функции сравнения (==, !=, >, <, >=, <=) 19

2.7.4. Обработчик сигналов (ds, alert) 20

2.7.4.1. Захват значения источника данных (ds) 20

2.7.4.2. Захват значения / статуса / длительности статуса сигнала (alert) 20

Предметный указатель 23

1.

Введение

1.1. Назначение ПМ PhoenixDS CmpEvnt

ПМ **PhoenixDS CmpEvnt** предназначен для создания сценариев дополнительной обработки данных, полученных от различных источников, с применением нестандартной логики. Может использоваться для агрегирования данных, а также для обработки сторонних данных при интеграции систем

2.

Настройка и использование ПМ PhoenixDS CmpEvnt

2.1. Устройство комплексной логики (Devices::Analysis::Logix)

Устройство комплексной логики (**Complex Logic**) предназначено для дополнительной обработки данных, полученных от различных источников, с применением нестандартной логики. Может использоваться для агрегирования данных, а также для обработки сторонних данных при интеграции систем.

Устройство комплексной логики состоит из одного целевого объекта с произвольным набором источников данных, настроенных администратором. Значение каждого источника данных вычисляется на основании построенной схемы или кода, введенного администратором при создании устройства. Поэтому для конфигурирования устройства комплексной логики с помощью кода администратор должен обладать базовым уровнем знания языка Perl.

Для получения данных с устройств комплексной логики в ПК **PhoenixDS Monitoring** необходимо запустить системный процесс `kollector-logix`.

2.2. Структура окна создания устройства

В верхней части окна создания устройства комплексной логики находится область, содержащая следующие параметры:

- **Отображаемое имя** | `display-name`;

  Имя объекта, отображаемое в дереве конфигурации. Должно быть уникальным в пределах одного уровня конфигурации (например, в рамках одной папки).

- **Описание** | `long_desc`.

  Развернутое описание объекта. Может содержать HTML-теги и пробелы.

- **Выбор коллектора** | `collector-name`

  Имя системного процесса, который собирает данные с устройства комплексной логики.

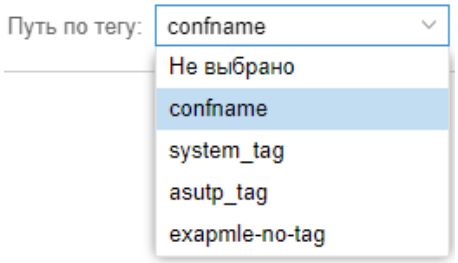
Под областью с параметрами находится область настройки сценариев, которая предназначена для создания схемы или написания программного кода на языке Perl для вычисления значений источников данных. Данная область содержит:

- панель со вкладками создаваемых сценариев (название вкладки можно отредактировать, нажав на иконку карандаша; также можно поменять порядок расположения вкладки, перетащив ее на место другой); с помощью кнопки на вкладке можно удалить сценарий, с помощью кнопки справа от конечной вкладки – добавить новый;
- панель инструментов, содержащую следующие элементы:

– имя вкладки, сохраняемое на сервере;

– поле для выбора цвета графика;

– кнопка для вставки шаблона кода выбранной функции;



– поле для выбора тега, в соответствии с которым определяется путь узла дерева конфигурации.

Если тег отсутствует или не выбран, подставляется полный путь узла.

- ❖ рабочую область вкладки для создания схемы / написания кода;
- ❖ панель выбора представления сценария (доступна для типа сценария **Схема**): **Схема** / **Текстовое представление** / **Просмотр тестового дерева**; подробнее см. п. Схема для вычисления значения источника данных 12.

В правой части окна расположена область с деревом конфигурации.

В нижней части окна расположена панель управления, содержащая кнопки для сохранения / отмены сделанных изменений.

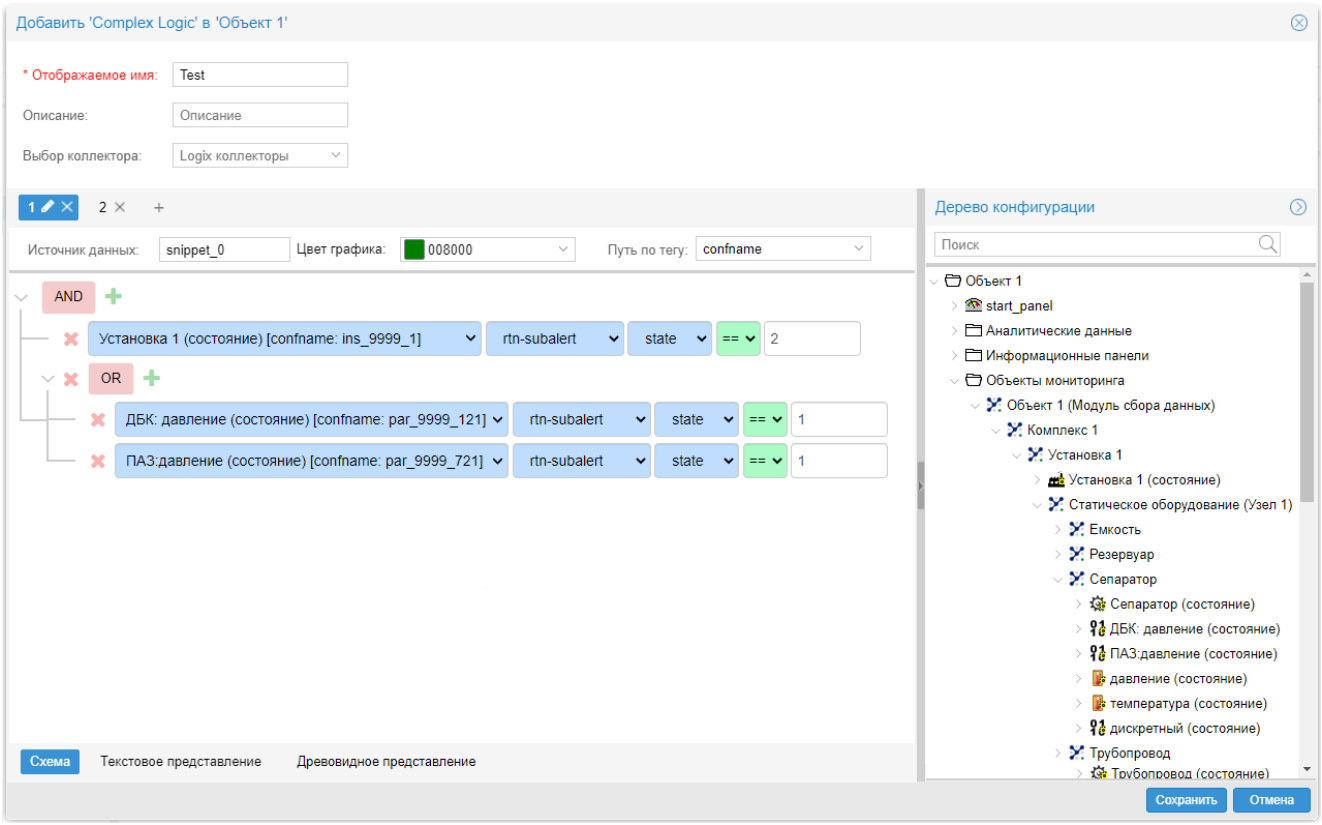


Рисунок 1. Окно создания устройства комплексной логики (тип сценария Схема)

2.3. Создание устройства

Общий порядок действий для создания нового устройства комплексной логики:

1. в дереве конфигурации поместить курсор на нужную папку, в контекстном меню выбрать пункт **Добавить устройство** и в появившемся списке выбрать **Complex Logic**;
2. в появившемся окне ввести параметры устройства **Отображаемое имя** и **Описание**;

3. нажать кнопку + на панели, предназначенной для отображения вкладок со сценариями; в появившемся окне ввести **Название сценария**, выбрать **Тип сценария** (**Код** / **Схема**) и нажать кнопку **Создать** (если выбран тип сценария **Схема**, то необходимо также указать первоначальную функцию узла и нажать кнопку **ОК**); в результате будет создана вкладка с введенным именем;
4. для вычисления значения источника данных на созданной вкладке ввести код на языке Perl или составить схему (в зависимости от выбраного типа сценария). Подробности см. ниже;
5. если для устройства комплексной логики требуется настроить несколько сценариев, то повторить п.п.3-4;
6. нажать кнопку **Сохранить**.

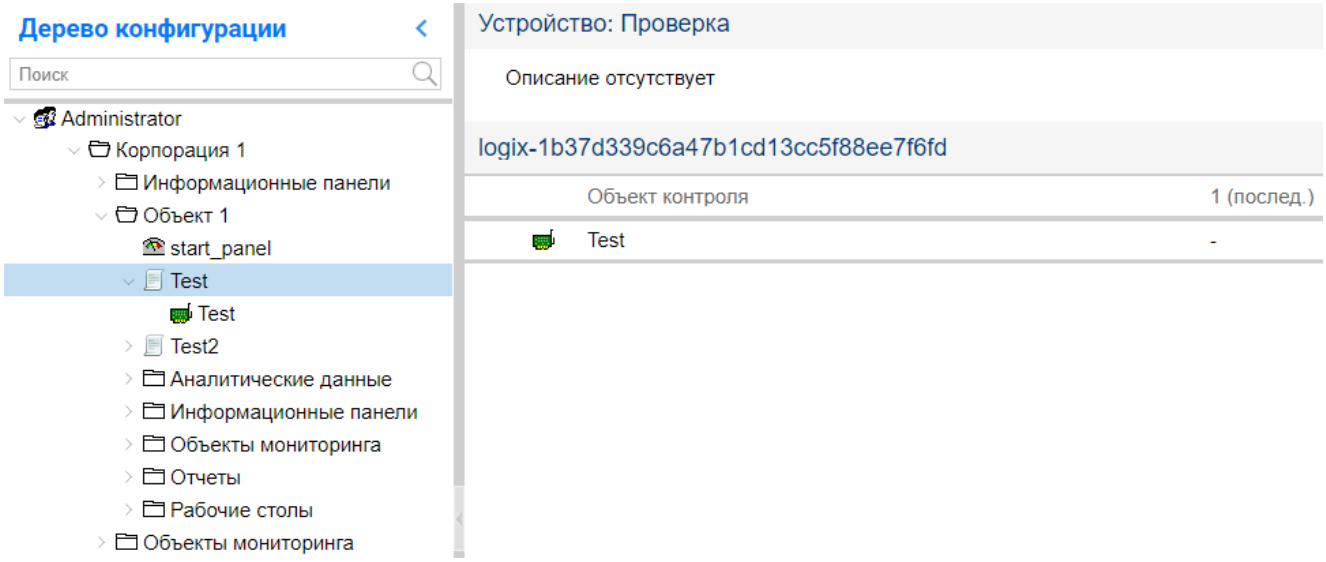


Рисунок 2. Устройство комплексной логики

2.4. Код для вычисления значения источника данных

При разработке кода на вкладках удобно пользоваться возможностями вставки шаблона кода. Для этого сначала в рабочей области вкладки поместить курсор в позицию для вставки шаблона кода, а затем воспользоваться одной из сервисных возможностей:

- вставка функции – нажать на панели инструментов кнопку f(x) и выбрать из списка функцию:
  - **Логические** – `AND`, `OR`, `ANY`, `NOT`, `NAND`, `NOR`, `XOR`;
  - **Временные** – `TON`, `TOF`, `STOF`;
  - **Сравнения** – `>`, `>=`, `<=`, `<`, `==`, `!=`;
  - **Дерево отказов** – `VOTE`, `INHIBIT`, `PAND`, `AND`, `OR`, `FDEP`, `SPARE`;
  - **Обработчик сигналов** – `ds`, `alert`.

  Подробное описание функций см в п. Функции 18.

  В результате в рабочую область будет вставлена заготовка кода для вызова функции. Останется отредактировать ее аргументы.
- вставка пути на объект дерева конфигурации или функций `alert` и `ds` с заданным аргументом:
  1. в поле **Путь по тегу** выбрать тег, в соответствии с которым будет определяться путь объекта дерева конфигурации: `confname`, `system_tag`, `asutp_tag`, `example-no-tag` (если тег не выбрать, то при перетаскивании узла по умолчанию подставится полный путь);
  2. в дереве конфигурации (справа от рабочей области вкладки с текстом кода) поместить курсор на нужный узел (в дереве отображаются, в том числе, источники данных целевых объектов);

3. нажатием ЛКМ перетащить узел в рабочую область вкладки. В результате в рабочую область добавляется путь или функция с заданным аргументом.

Примеры.

Пример 1. Если в дереве конфигурации выбрано устройство, то в текстовое поле вставляется путь к узлу.

Рисунок 3. Вставка пути на объект дерева конфигурации

Пример 2. Если в дереве конфигурации выбран набор сигналов, то в текстовое поле вставляется код вызова функции `alert` с указанием в качестве аргументов выбранного набора сигналов и первого сигнала этого набора.

Рисунок 4. Вставка функции alert

В данном примере в поле **Путь по тегу** указан тег `confname`, поэтому в текстовое поле подставилось название функции (**getPathByUniqueTag**), с помощью которой происходит поиск пути по тегу, название тега (**confname**) и его значение (**ins_9999_1**).

Пример 3. Если в дереве конфигурации выбран источник данных, то в текстовое поле вставляется код вызова функции `ds` с указанием в качестве аргумента выбранного источника данных.

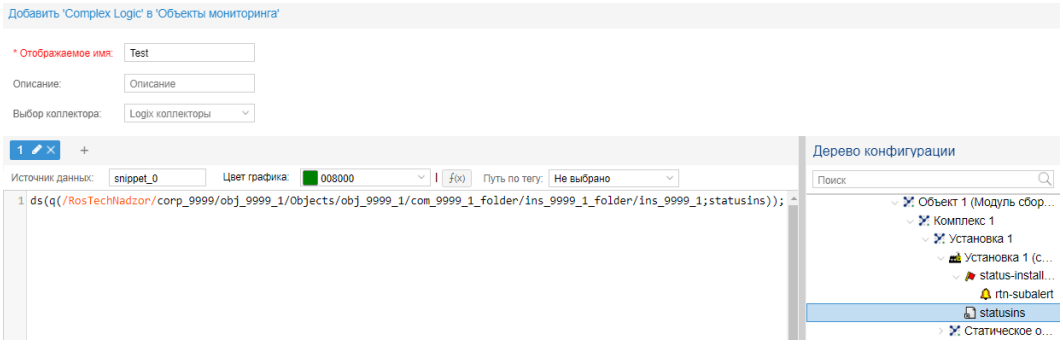


Рисунок 5. Вставка функции ds

2.5. Схема для вычисления значения источника данных

Тип сценария **Схема** предполагает отображение сценария в 3-х представлениях:

- **Схема** – в виде графической иерархии элементов; см. п. Представление Схема 12;
- **Текстовое представление** – в текстовом формате json; см. п. Текстовое представление 16;
- **Древовидное представление** – в виде тестового дерева, заполненного случайными параметрами; см. п. Древовидное представление 17.

Панель с кнопками для выбора представления расположена под рабочей областью настройки сценариев.

Редактировать сценарий можно как в представлении **Схема**, так и в **Текстовом представлении**. Данные представления синхронизированы друг с другом (изменения, внесенные в сценарий в одном представлении, автоматически отображаются во втором).

Сценарий в **Древовидном представлении** редактировать нельзя, однако изменения, сделанные в двух других представлениях, будут также отображены в данном представлении.

Из дерева конфигурации в рабочую область вкладки можно перетащить узел, к которому привязан шаблон сигналов, его шаблон сигналов и сигнал, но при условии, что этот узел или его родитель обладают выбранным тегом. В **Текстовом представлении** отсутствует возможность перетаскивания в рабочую область узлов дерева конфигурации.

При неправильном введении текстового сценария выдается предупреждение о невозможности преобразования текстового сценария в схему, и предлагается либо исправить в **Текстовом представлении** ошибку сценария, либо отменить изменения сценария.

2.5.1. Представление Схема

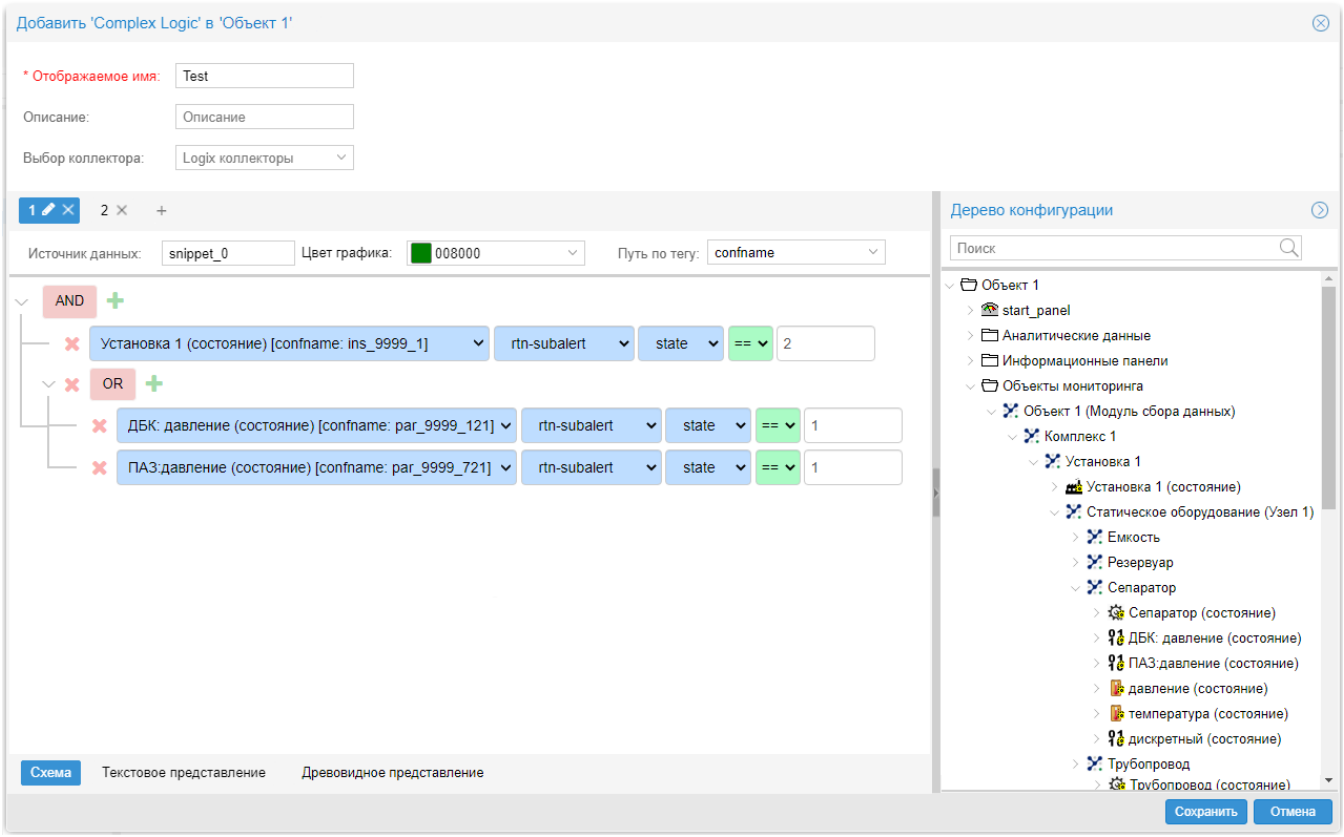


Рисунок 6. Представление Схема

При создании схемы в появившемся окне необходимо выбрать первоначальную функцию узла. Если функция не выбрана, по умолчанию используется функция `AND`. При необходимости можно заменить выбранные функции на другие в пределах их групп.

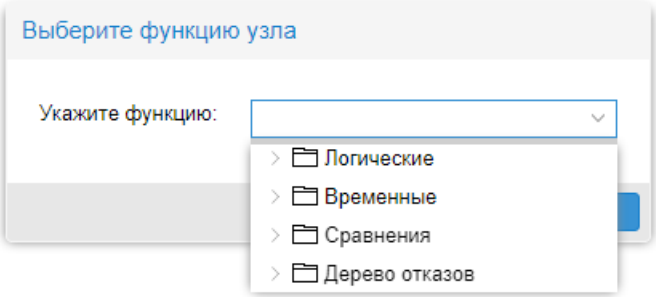


Рисунок 7. Окно для выбора функции узла

Кнопка + позволяет добавлять функции или правила, кнопка ✖ – удалять, кнопки ⌄ / › позволяют сворачивать / разворачивать группу правил. При наведении мыши на правило оно подсвечивается, и рядом с ним появляется кнопка ↱, позволяющая перейти на соответствующий правилу целевой объект в дереве конфигурации.

2.5.1.1. Добавление функции/правила в сценарий

Добавление функции в сценарий

Добавление функции осуществляется с помощью кнопки +, которая помещает ее на уровень ниже по отношению к выбранной функции.

Чтобы добавить функцию в сценарий, следует:

1. нажать кнопку + рядом с функцией, для которой необходимо создать дочернюю функцию;
2. в открывшемся меню выбрать пункт **Добавить функцию**;

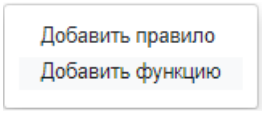


Рисунок 8. Меню добавления функции или правила

3. в открывшемся окне выбрать функцию из предложенных типов и нажать кнопку **ОК**.

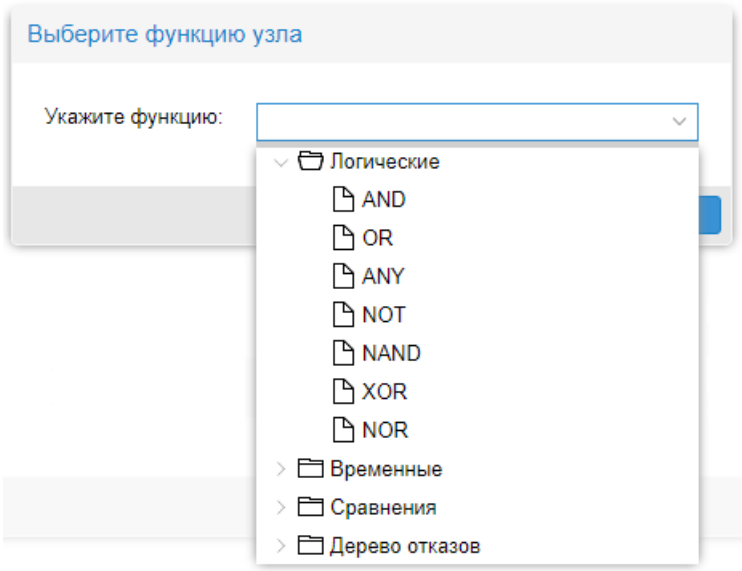


Рисунок 9. Меню выбора функции

Подробное описание функций см. в п. Функции 18.

В результате функция добавится в следующий уровень иерархии. Если в окне выбора функции узла при нажатии кнопки **ОК** функция не указана, по умолчанию добавляется функция `AND`.

Существует возможность замены выбранной функции другой такого же типа. Для этого следует нажать на кнопку функции, которую необходимо изменить, например AND, и в открывшемся меню выбрать необходимую функцию, после чего на месте кнопки отобразится соответствующее название.

Добавление правила в сценарий

Чтобы добавить правило в сценарий, следует:

1. выбрать узел с функцией, в которую необходимо добавить правило, и нажать кнопку ;
2. в открывшемся меню выбрать пункт **Добавить правило**;

В результате правило добавится в конец группы правил выбранной функции.

Также можно добавить правило, перетащив в рабочую область редактора узел дерева конфигурации, к которому привязан шаблон сигналов и который обладает выбранным тегом, или его шаблон сигналов, или сигнал.

2.5.1.2. Настройка правила

Чтобы настроить правило, необходимо настроить его поля. Для этого следует:

1. в первом поле при необходимости изменить целевой объект, сигнал которого необходимо настроить (в выпадающем списке отображаются названия целевых объектов, уже использованных в сценарии);

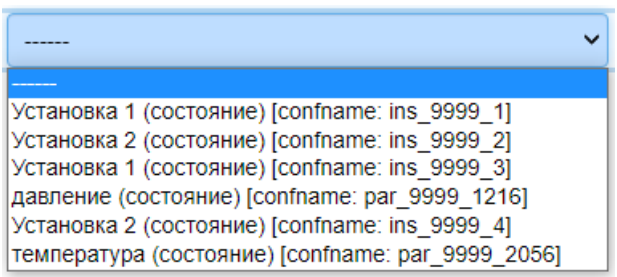


Рисунок 10. Поле для изменения целевого объекта, сигнал которого необходимо настроить

2. во втором поле из списка выбрать сигнал (в случае, если шаблон сигналов содержит несколько сигналов);

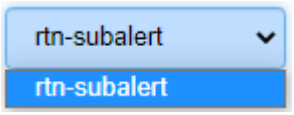


Рисунок 11. Поле для выбора сигнала

3. в третьем поле указать параметр, который будет анализироваться в правиле:
   - **state** – статус сигнала;
   - **value** – значение сигнала;
   - **duration** – длительность последнего статуса сигнала;

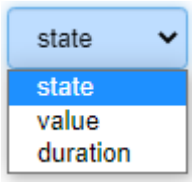


Рисунок 12. Поле для выбора анализируемого параметра

4. в четвертом поле выбрать оператор сравнения с некоторым значением (количество операторов в списке зависит от параметра, выбранного в третьем поле);

Рисунок 13. Поле для выбора оператора сравнения

5. в пятом поле ввести значение, с которым будет сравниваться значение параметра, выбранного в третьем поле (при необходимости ввести несколько значений через запятую).

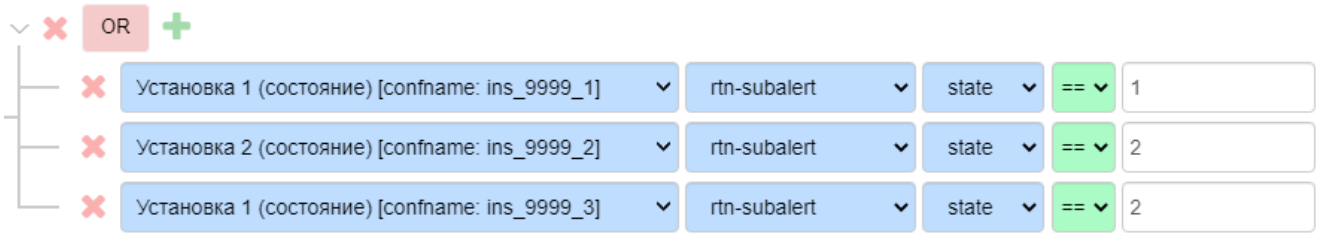


Рисунок 14. Настройка правил в группе

Для некоторых функций необходимо настроить обязательные поля. Например, для логической функции `ANY` настраивается значение поля **Количество**, для временных функций `TON`, `TOF`, `STOF` настраивается значение поля **Длительность**.

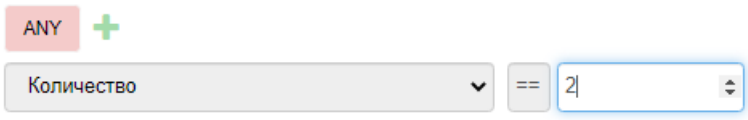


Рисунок 15. Обязательное поле для логической функции ANY

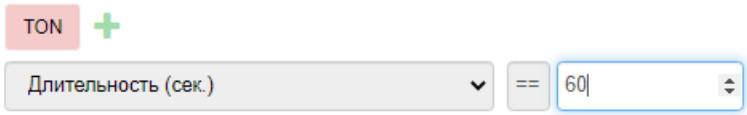


Рисунок 16. Обязательное поле для временной функции (пример для функции TON)

2.5.2. Текстовое представление

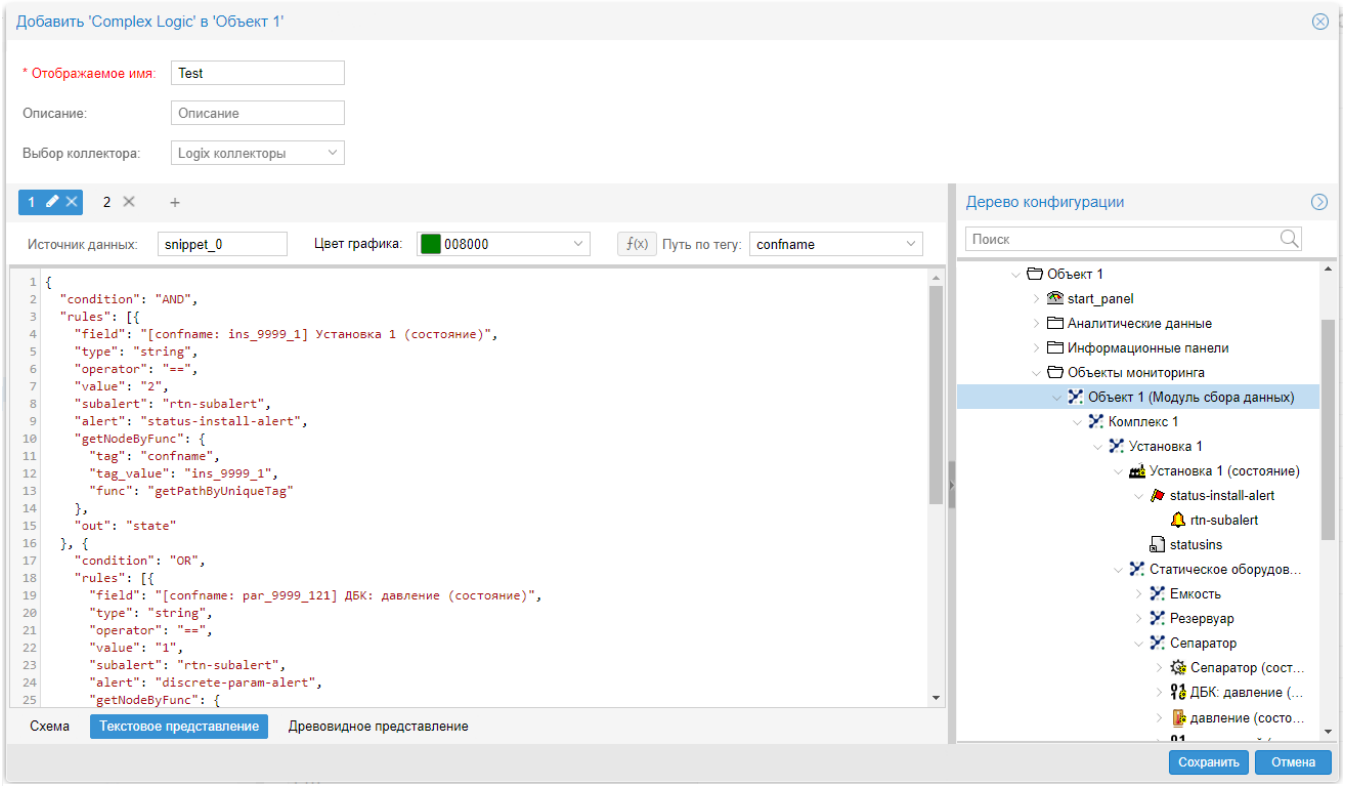


Рисунок 17. Текстовое представление

Кнопка позволяет вставить в рабочую область редактора шаблон кода выбранной функции.

2.5.3. Древовидное представление

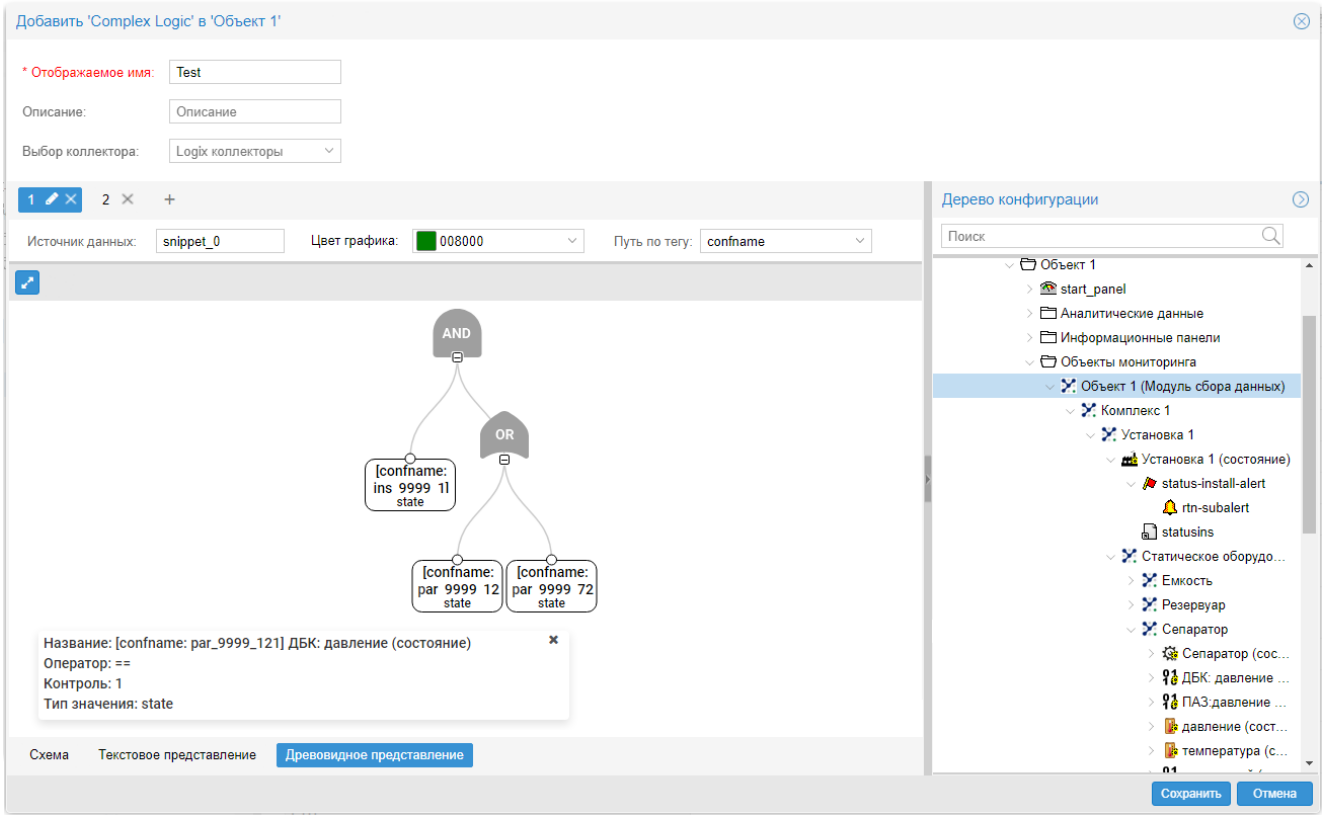


Рисунок 18. Древовидное представление

Кнопка / позволяет свернуть / развернуть во весь экран дерево.

При нажатии на узел с функцией его дочерние элементы сворачиваются, при повторном нажатии снова отображаются.

При нажатии на узел с параметром в нижней части рабочей области отображается подсказка с информацией об этом узле.

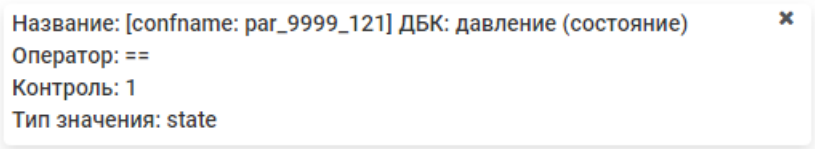


Рисунок 19. Пример подсказки с информацией об узле

2.6. Удаление сценария с устройства комплексной логики

Для удаления сценария с устройства следует в окне редактирования параметров устройства перейти на вкладку со сценарием, который нужно удалить, и нажать кнопку рядом с названием сценария.

В результате вкладка с выбранным сценарием будет удалена. Для сохранения изменений следует нажать в нижней части окна на панели управления кнопку **Сохранить**.

2.7. Функции

В настоящем разделе описаны функции, которые могут использоваться в коде / схеме вычисления значения источника данных для устройства комплексной логики.

2.7.1. Логические функции (AND, OR, ANY, NOT, NAND, NOR, XOR)

Функция `AND` проверяет, что все M переданных в функцию правил выполняются. Для корректной работы необходимо выполнение условия M > 0.

Функция `OR` проверяет, что хотя бы 1 из M переданных в функцию правил выполняется. Для корректной работы необходимо выполнение условия M > 0.

Функция `ANY` проверяет наличие любых N выполненных правил среди M переданных в функцию правил. Для корректной работы необходимо выполнение условия M >= N.

Функция `NOT` проверяет, что переданное в функцию правило не выполняется.

Функция `NAND` проверяет, что хотя бы одно из M переданных в функцию правил не выполняется. Для корректной работы необходимо выполнение условия M > 0.

Функция `NOR` проверяет, что все M переданных в функцию правил не выполняются. Для корректной работы необходимо выполнение условия M > 0.

Функция `XOR` проверяет, есть ли нечетное количество выполненных правил среди M правил, переданных в функцию. Для корректной работы необходимо выполнение условия M > 0.

Синтаксис

Функция `AND`:

```
AND(rule(s))
```

Функция `OR`:

```
OR(rule(s))
```

Функция `ANY`:

```
ANY(min,rule(s))
```

Функция `NOT`:

```
NOT(rule)
```

Примеры

Пример использования функции `AND`:

```
AND(alert(q(path_to_device;alert-template;alert))-
>state==3,alert(q(path_to_device;alert-template;alert))->state==3)
```

Пример использования функции `OR`:

```
OR(alert(q(path_to_device;alert-template;alert))-
>state==3,alert(q(path_to_device;alert-template;alert))->state==3)
```

Пример использования функции `ANY`:

```
ANY(2,alert(q(path_to_device;alert-template;alert))-
>state==3,alert(q(path_to_device;alert-template;alert))->state==3)
```

Пример использования функции NOT:

```
NOT(alert(q(path_to_device;alert-template;alert))->state==3)
```

2.7.2. Временые функции (TON, TOF, STOF)

Функция TON проверяет, что переданное в функцию правило выполняется в течение заданного периода времени.

Функция TOF проверяет, что переданное в функцию правило было выполнено в течение заданного времени в прошлом.

Функция STOF проверяет, что после реализации первого переданного в функцию правила в течение заданного времени выполняется второе правило. Второе событие является следствием первого события.

Синтаксис

Функция TON:

```
TON(Timeout,rule)
```

Функция TOF:

```
TOF(Timeout,rule)
```

Функция STOF:

```
STOF(rule,Timeout,rule)
```

Примеры

Пример использования функции TON:

```
TON(120,alert(q(path_to_device;alert-template;alert))->state==3)
```

Пример использования функции TOF:

```
TOF(120,alert(q(path_to_device;alert-template;alert))->state==3)
```

Пример использования функции STOF:

```
STOF(alert(q(path_to_device;alert-template;alert))-
>state==3,120,alert(q(path_to_device;alert-template;alert))->state==3)
```

2.7.3. Функции сравнения (==, !=, >, <, >=, <=)

Функция == сравнивает два значения и возвращает истину, если значения равны, функция != – если значения не равны, функция > – если первое значение больше второго, функция < – если первое значение меньше второго, функция >= – если первое значение больше или равно второму, функция <= – если первое значение меньше или равно второму.

Для корректной работы функций необходимо указать два выражения, значения которых будут сравниваться.

Функция	Расшифровка
==	Равно
!=	Не равно
>	Больше

<	Меньше
>=	Больше или равно
<=	Меньше или равно

Синтаксис

```
<функция сравнения>(rule,rule)
```

`rule` – выражение, значение которого будет сравниваться.

2.7.4. Обработчик сигналов (ds, alert)

2.7.4.1. Захват значения источника данных (ds)

Функция `ds` возвращает поступившие по шине ESB от коллекторов значения (или одно значение) от источников данных. Аргументом функции является источник данных или поддерево.

Синтаксис

```
ds(q(<path_to_ds>))
```

```
<path_to_ds>::=<Путь к таргету>[;<Имя источника данных>]
```

`<Путь к таргету>` – существующий путь в дереве конфигурации (как в `/Devices`, так и в `/Tree` и т.п.) к целевому объекту (например: `/Tree/SelfMon/localhost/cpu`) или поддереву (например: `/Tree/devs/ws_040`). В последнем случае результат будет векторным – множество значений всех источников данных в поддереве.

`<Имя источника данных>` – имя источника данных (например: `sysPercentProcessorLoad`). Необязательный аргумент. Если задано имя источника, а путь задан для поддерева, то результатом функции будет множество значений всех источников данных с таким именем, имеющихся в поддереве. Если имя источника данных не задано, то результатом функции будет множество значений всех источников в целевом объекте (или поддереве).

Пример

Захват значений загрузки `sysPercentProcessorLoad` по каждому процессору устройства `/Tree/devs/ws_040` (результат – множество значений):

```
ds(q(/Tree/devs/ws_040;sysPercentProcessorLoad))
```

2.7.4.2. Захват значения / статуса / длительности статуса сигнала (alert)

В зависимости от заданного аргумента функция `alert` возвращает значение / статус / длительность последнего статуса одного сигнала либо множество значений или статусов сигналов. Под статусами сигналов подразумеваются соответствующие им числовые константы: 0 – синий, 1 – зеленый, 2 – желтый, 3 – красный, 4 – бордовый, 5 – серый. Аргументом функции является путь к сигналу конкретного целевого объекта или ветви дерева конфигурации.

Синтаксис

```
alert(q(<path_to_alert>))->{state|value|duration}
```

```
<path_to_alert>::=<Путь к таргету>[;<Имя шаблона сигналов>];[<Имя сигнала>]
```

<Путь к таргету> – любой существующий путь в дереве конфигурации (только в /Tree) к целевому объекту или поддереву. См. также аналогичный атрибут для функции ds.

<Имя шаблона сигналов> – имя набора сигналов, например: lnx-cpu; будут рассматриваться только целевые объекты, которым назначен данный набор сигналов. Необязательный атрибут. Если он не задан, то используется любой обнаруженный набор сигналов.

<Имя сигнала> – имя сигнала, например: load; будет рассматриваться только данный сигнал. Необязательный атрибут. Если он не задан, то используются все сигналы (результат будет векторным).

{state|value|duration}

Тип возвращаемого значения:

state – статус сигнала или множество статусов сигналов;

value – значение сигнала или множество значений сигналов;

duration – длительность последнего статуса сигнала.

Примеры

Получение статусов сигналов load по всем процессорам устройства ws_040:

```
alert(q(/Tree/devs/ws_040;load))->state
```

Результат выполнения функции – множество числовых констант, соответствующих статусам сигналов.

Получение значений всех сигналов из набора lnx-storage, привязанного к целевым объектам устройства localhost:

```
alert(q(/Tree/SelfMon/localhost;lnx-storage;))->value
```

Получение значений всех сигналов всех наборов сигналов в дереве /Tree:

```
alert(q(/Tree;))->value
```

Предметный указатель

K

kollector 8

И

Источник данных 8, 20

Т

Тег 8, 10