

Модуль "XML parser"
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Содержание

Раздел 1 Введение	1
Раздел 2 Системные требования	1
Раздел 3 Установка XML parser	1
Раздел 4 Глоссарий	2
Раздел 5 Руководство пользователя	3
1 Анализ и разбор данных	3
2 Узлы XML	4
3 Замена символов	6
4 Фильтр	7
5 Синтаксис регулярных выражений	8
6 Форматирование данных	13
Раздел 6 Проблемы?	14
1 Возможные проблемы	14

1 Введение

Этот модуль позволяет извлекать данные формата XML из потока данных. Сначала парсер ищет в потоке заданный корневой узел, а затем автоматически извлекает значения вложенных узлов и атрибутов в переменные.

Подготовленные данные (переменные) передаются в другие модули (например, модули экспорта данных) в виде переменных, содержащих извлеченные значения.

2 Системные требования

Для установки XML parser должны быть выполнены следующие требования:

Операционная система: Windows 2000 SP4 и выше, включая версии для 32-х и 64-х битных систем, а также серверные версии операционных систем.
Наличие установленного последнего сервис-пака желательно.

Свободное дисковое пространство: Рекомендуется не менее 5 МВ свободного дискового пространства.

Специальные требования для доступа: Вам потребуются права Администратора для установки модуля. Для некоторых модулей могут потребоваться права администратора для выполнения заданных функций.

Необходимо наличие главного приложения (ядра), например Advanced Serial Data Logger.

3 Установка XML parser

1. Закройте главное приложение (например, Advanced Serial Data Logger), если оно запущено;
2. Скопируйте программу на ваш жесткий диск;
3. Запустите программу установки модуля, кликнув два раза на имени файла в проводнике Windows;
4. Следуйте инструкциям программы установки. Обычно достаточно нажать несколько раз кнопку "Дальше";
5. Запустите главное приложение. В случае успешной установки название модуля появится в окне настройки, на закладке "Модули".

Если модуль совместим с программой, то его название и номер версии будет отображаться в списке модулей. Примеры установленных модулей можно посмотреть на рис. 1-2. Некоторые типы модулей требуют дополнительной настройки. Для этого достаточно выбрать модуль из списка и нажать кнопку "Настроить" рядом со списком. Процедура настройки модуля описана в следующих главах.

Некоторые типы модулей видны на закладке "Лог-файл". Для их настройки необходимо выбрать модуль из списка "Тип файла" и нажать кнопку "Дополнительно".

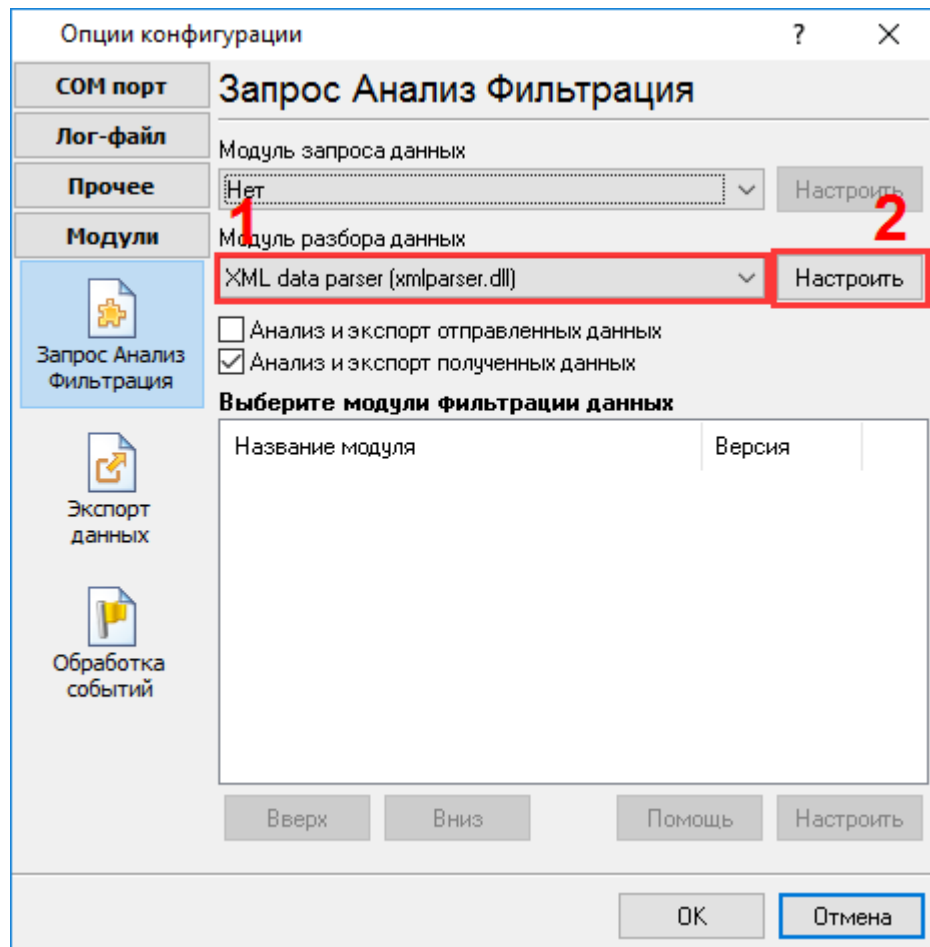


Рис. 1. Пример установленного модуля

4 Глоссарий

Plug-in - модуль

Главная программа - программная оболочка, которая использует данный модуль. Например: Advanced Serial Data Logger

Основная программа - см. "Главная программа".

Парсер - модуль, который обрабатывает поток данных, выделяя из него пакеты данных и переменные из пакетов данных. Затем эти переменные используются в модулях экспорта данных.

Ядро - см. "Главная программа".

5 Руководство пользователя

5.1 Анализ и разбор данных

Для того чтобы экспортировать полученные из порта данные, вам необходимо произвести конфигурацию парсера. Вы должны настроить правила, по которым парсер будет извлекать данные из потока на закладке "Данные XML" (рис.1).

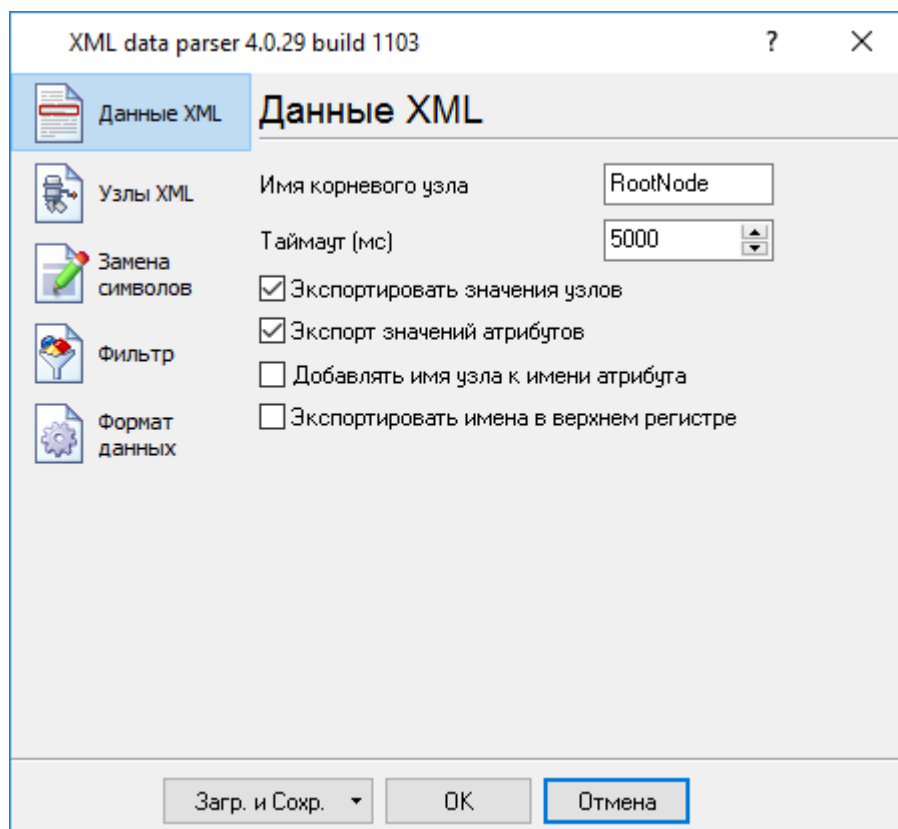


Рис.1. Данные XML.

Имя корневого узла – имя корневого узла XML (регистр символов не учитывается), который содержит экспортируемые данные. Например, Response.

Модуль будет извлекать данные из пакета вида:

```
<Response end="#0A" parse="bp" timeout="500" validatesize="0">
  <Item type="filter" size="-1" filtertype="1" exprtype="1" expr="----" />
  <Item type="filter" size="-1" filtertype="2" exprtype="2" expr="^\d{2}\|
\d{2}\|\d{2} " />
</Response>
```

Все узлы внутри <Response> и </Response>, и все атрибуты узла <Response> будут обработаны.

Если узел XML не содержит завершающего тега (</Response>), то этот кусок XML не будет обработан. Например, следующие примеры верных XML данных НЕ будут обработаны.

```
<Response end="#0A" parse="bp" timeout="500" validatesize="0">
...
</Response end="1">
```

```
<Response end="#0A" parse="bp" timeout="500" validatesize="0" />
```

Таймаут – модуль будет ожидать данные для указанного корневого узла в течении данного временного интервала.

Экспортировать значения узлов – если данная опция включена, то модуль будет извлекать значения вида <Item>Значение</Item>;

Экспорт значений атрибутов – если данная опция включена, то модуль будет обрабатывать все атрибуты узла.

Пример: <Response end="#0A" parse="bp" timeout="500" validatesize="0">;

в данном примере атрибуты end, parse, timeout, validatesize будут обработаны.

Добавлять имя узла к имени атрибута – если данная опция включена, то итоговое имя переменной парсера будет состоять из названия узла и имени атрибута. Для примера, приведенного выше, атрибуты end, parse, timeout, validatesize будут экспортированы с именами Response.end, Response.parse, Response.timeout, Response.validatesize. Это позволяет в дальнейшем отделить атрибуты разных узлов, но с одинаковыми именами.

Экспортировать имена в верхнем регистре – если данная опция включена, то имена всех узлов и атрибутов будут преобразованы в заглавные буквы.

5.2 Узлы XML

Список на закладке "Узлы XML" (рис. 2) позволяет задать тип обработки и тип данных для узлов и атрибутов XML. Вы также можете указать, какие узлы и атрибуты нужно игнорировать. Кликните на кнопке "Добавить" и добавьте нужные правила в список.

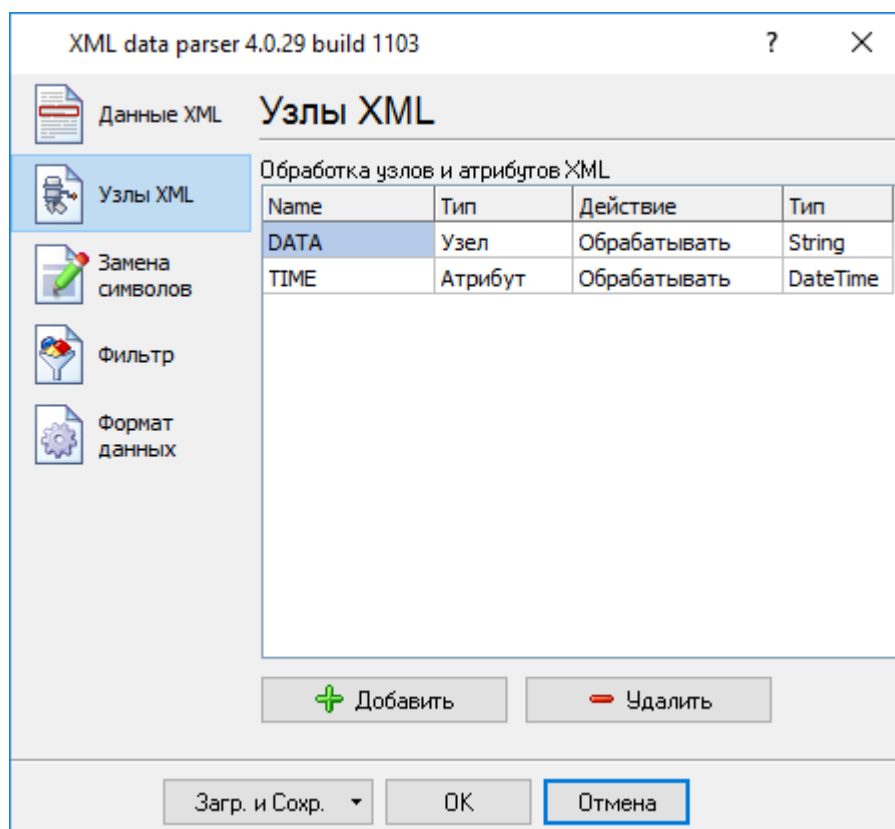


Рис.2. Узлы XML.

Имя - имя узла или атрибута (без учета регистра символов).

Тип - тип правила. Определяет к какой части XML будет данное правило применяться: Все, Узел или Атрибут.

Действие

- **Игнорировать** – указанный узел или атрибут будет проигнорирован;
- **Обрабатывать** – указанный узел или атрибут будет обработан.

Типы данных

- String - строковое значение - Набор символов длиной от 1 до 65535 символов;
- Memo - строковое значение - Набор символов длиной от 1 до 2³² символов;
- Bytes - двоичное значение;
- Blob - Binary Large Object field (набор байт);
- Boolean - Логическое значение (True/False) - 0 или 1;
- Float - действительное число - диапазон значений: -2.9 x 10⁻³⁹ .. 1.7 x 10³⁸
- Smallint - знаковое целое число - диапазон: 32768..32767;
- Word - беззнаковое целое число - диапазон: 0..65535;
- Integer - знаковое целое число: -2147483648..2147483647;
- Date - дата;
- Time - время;
- DateTime - дата и время.

5.3 Замена символов

Замена символов (рис.3) используется, когда вы хотите удалить или заменить некоторые символы из пакета данных. Например, удалить непечатаемые символы ASCII.

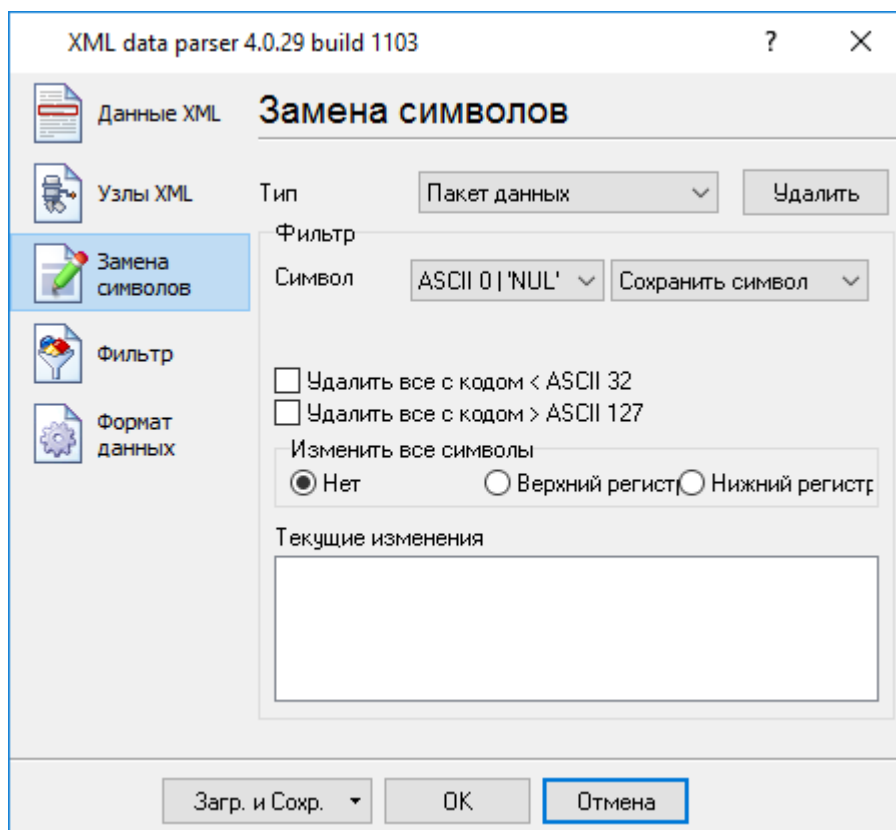


Рис.3 Замена символов

5.4 Фильтр

Фильтр предназначен для того, чтобы игнорировать некоторые пакеты данных, которые вы не хотите экспортировать с помощью других модулей.

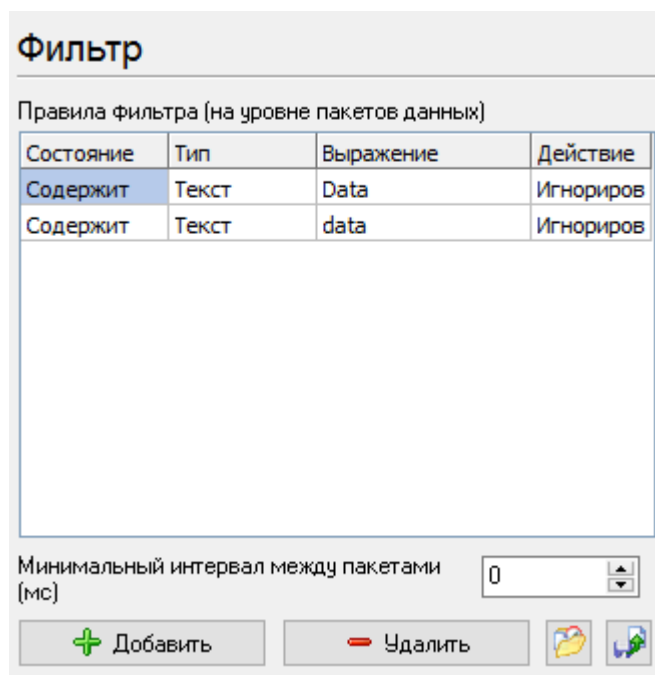


Рис. 2. Правила фильтрации

Для фильтрации необходимо определить одно или несколько правил. Если какое-либо правило выполняется, то с пакетом производится действие, указанное в поле "Действие".

Типы действий

- **Игнорировать** - текущий пакет данных будет проигнорирован и не будет экспортироваться;
- **Обрабатывать** - текущий пакет данных будет обработан и экспортирован.

Существует несколько типов условий, которые указываются в поле "Состояние".

Типы состояний правила

- **Отключено** - это правило отключено и не используется для фильтрации;
- **Содержит** - это правило проверяет наличие строки/выражения из поля "Выражение" в пакете данных;
- **Не содержит** - это правило проверяет отсутствие строки/выражения из поля "Выражение" в пакете данных;

Типы выражений - Выражения в поле "Выражение" могут быть 2х типов:

Текст - модуль будет искать в пакете данных строку, указанную в поле "Выражение". Поиск зависит от регистра символов.

Рег. выраж. - модуль будет использовать для поиска [регулярное выражение](#), указанное в поле "Выражение". Поиск зависит от регистра символов.

5.5 Синтаксис регулярных выражений

Регулярные выражения - мощный инструмент, который должен поможет находить совпадения в строках на основе довольно сложных шаблонов.

Квантификаторы

	Аналог	Пример	Описание
?	{0,1}	a?	одно или ноль вхождений "a".
+	{1,}	a+	одно или более вхождений "a".
*	{0,}	a*	ноль или более вхождений "a".

Модификаторы

Символ "минус" (-) перед модификатором (за исключением U) создаёт его отрицание.

	Описание
g	глобальный поиск (обрабатываются все совпадения с шаблоном поиска)
i	игнорировать регистр
m	многострочный поиск. Поясню: по умолчанию текст это одна строка, с модификатором есть отдельные строки, а значит ^ начало строки в тексте, \$- конец строки в тексте.
s	текст воспринимается как одна строка, спец символ "точка. (.)" будет включать и перевод строки
u	используется кодировка UTF-8
U	инвертировать жадность
x	игнорировать все неэкранированные пробельные и перечисленные в классе символы

Спецсимволы

	Аналог	Описание
()		подмаска, вложенное выражение
[]		групповой символ
{a,b}		количество вхождений от "a". до "b".
		логическое "или.", в случае с односимвольными альтернативами используйте []
\		экранирование спец символа
.		любой символ, кроме перевода строки
\d	[0-9]	десятичная цифра
\D	[^d]	любой символ, кроме десятичной цифры
\f		конец (разрыв) страницы
\n		перевод строки
\pL		буква в кодировке UTF-8 при использовании модификатора u
\r		возврат каретки

<code>\s</code>	<code>[\t\r\n\f]</code>	пробельный символ
<code>\S</code>	<code>[^\s]</code>	любой символ, кроме промельного
<code>\t</code>		табуляция
<code>\w</code>	<code>[0-9a-z_]</code>	любая цифра, буква или знак подчеркивания
<code>\W</code>	<code>[^\w]</code>	любой символ, кроме цифры, буквы или знака подчеркивания
<code>\v</code>		вертикальная табуляция

Спецсимволы внутри символьного класса

	Пример	Описание
<code>^</code>	<code>[^da]</code>	отрицание, любой символ кроме "d. или "a.
<code>-</code>	<code>[a-z]</code>	интервал, любой симво от "a. до "z.

Позиция внутри строки

	Пример	Соответствие	Описание
<code>^</code>	<code>^a</code>	aaa aaa	начало строки
<code>\$</code>	<code>a\$</code>	aaa aaa	конец строки
<code>\A</code>	<code>\Aa</code>	aaa aaa aaa aaa	начало текста
<code>\z</code>	<code>a\z</code>	aaa aaa aaa aaa	конец текста
<code>\b</code>	<code>a\b</code> <code>\ba</code>	aaa aaa aaa aaa	граница слова, утверждение: предыдущий символ словесный, а следующий - нет, либо наоборот
<code>\B</code>	<code>\Ba\B</code>	aaa aaa	отсутствие границы слова
<code>\G</code>	<code>\Ga</code>	aaa aaa	Предыдущий успешный поиск, поиск остановился на 4-й позиции - там, где не нашлось a

Якоря

Якоря в регулярных выражениях указывают на начало или конец чего-либо. Например, строки или слова. Они представлены определенными символами. К примеру, шаблон, соответствующий строке, начинающейся с цифры, должен иметь следующий вид:

```
^[0-9]+
```

Здесь символ `^` обозначает начало строки. Без него шаблон соответствовал бы любой строке, содержащей цифру.

Символьные классы

Символьные классы в регулярных выражениях соответствуют сразу некоторому набору символов. Например, `\d` соответствует любой цифре от 0 до 9 включительно, `\w` соответствует буквам и цифрам, а `\W` - всем символам, кроме букв и цифр. Шаблон, идентифицирующий буквы, цифры и пробел, выглядит так:

```
\w\s
```

Кванторы

Кванторы позволяют определить часть шаблона, которая должна повторяться несколько раз подряд. Например, если вы хотите выяснить, содержит ли документ строку из от 10 до 20 (включительно) букв "a.", то можно использовать этот шаблон:

```
a{10,20}
```

По умолчанию кванторы - "жадные". Поэтому квантор `+`, означающий "один или больше раз.", будет соответствовать максимально возможному значению. Иногда это вызывает проблемы, и тогда вы можете сказать квантору перестать быть жадным (стать "ленивым."), используя специальный модификатор. Посмотрите на этот код:

```
".*"
```

Этот шаблон соответствует тексту, заключенному в двойные кавычки. Однако, ваша исходная строка может быть вроде этой:

```
<a href="helloworld.htm" title="Привет, Мир">Привет, Мир</a>
```

Приведенный выше шаблон найдет в этой строке вот такую подстроку:

```
"helloworld.htm" title="Привет, Мир"
```

Он оказался слишком жадным, захватив наибольший кусок текста, который смог.

```
".*?"
```

Этот шаблон также соответствует любым символам, заключенным в двойные кавычки. Но ленивая версия (обратите внимание на модификатор `?`) ищет наименьшее из возможных вхождений, и поэтому найдет каждую подстроку в двойных кавычках по отдельности:

```
"helloworld.htm" "Привет, Мир"
```

Экранирование в регулярных выражениях

Регулярные выражения используют некоторые символы для обозначения различных частей шаблона. Однако, возникает проблема, если вам нужно найти один из таких символов в строке, как обычный символ. Точка, к примеру, в регулярном выражении обозначает "любой символ, кроме переноса строки". Если вам нужно найти точку в строке, вы не можете просто использовать `.` в качестве шаблона - это приведет к нахождению практически всего. Итак, вам необходимо сообщить парсеру, что эта точка должна считаться обычной точкой, а не "любимым символом". Это делается с помощью знака экранирования.

Знак экранирования, предшествующий символу вроде точки, заставляет парсер игнорировать его функцию и считать обычным символом. Есть несколько символов, требующих такого

экранирования в большинстве шаблонов и языков. Вы можете найти их в правом нижнем углу шпаргалки ("Мета-символы.).

Шаблон для нахождения точки таков:

\.

Другие специальные символы в регулярных выражениях соответствуют необычным элементам в тексте. Переносы строки и табуляции, к примеру, могут быть набраны с клавиатуры, но вероятно собьют с толку языки программирования. Знак экранирования используется здесь для того, чтобы сообщить парсеру о необходимости считать следующий символ специальным, а не обычной буквой или цифрой.

Спецсимволы экранирования в регулярных выражениях

Выражение	Соответствие
\	не соответствует ничему, только экранирует следующий за ним символ. Это нужно, если вы хотите ввести метасимволы <code>!\$()*+.<>?[\]^{ }</code> в качестве их буквальных значений.
\Q	не соответствует ничему, только экранирует все символы вплоть до \E
\E	не соответствует ничему, только прекращает экранирование, начатое \Q

Группы и диапазоны

Группы и диапазоны очень-очень полезны. Вероятно, проще будет начать с диапазонов. Они позволяют указать набор подходящих символов. Например, чтобы проверить, содержит ли строка шестнадцатеричные цифры (от 0 до 9 и от A до F), следует использовать такой диапазон:

```
[A-Fa-f0-9]
```

Чтобы проверить обратное, используйте отрицательный диапазон, который в нашем случае подходит под любой символ, кроме цифр от 0 до 9 и букв от A до F:

```
[^A-Fa-f0-9]
```

Группы наиболее часто применяются, когда в шаблоне необходимо условие "или."; когда нужно сослаться на часть шаблона из другой его части; а также при подстановке строк.

Использовать "или." очень просто: следующий шаблон ищет "ab." или "bc.":

```
(ab|bc)
```

Если в регулярном выражении необходимо сослаться на какую-то из предшествующих групп, следует использовать \n, где вместо \n подставить номер нужной группы, например \1. Вам может понадобиться шаблон, соответствующий буквам "aaa." или "bbb.", за которыми следует число, а затем те же три буквы. Такой шаблон реализуется с помощью групп:

```
(aaa|bbb)[0-9]+\1
```

Первая часть шаблона ищет "aaa." или "bbb.", объединяя найденные буквы в группу. За этим следует поиск одной или более цифр ([0-9]+), и наконец \1. Последняя часть шаблона

ссылается на первую группу и ищет то же самое. Она ищет совпадение с текстом, уже найденным первой частью шаблона, а не соответствующее ему. Таким образом, "aaa123bbb" не будет удовлетворять вышеприведенному шаблону, так как \1 будет искать "aaa" после числа.

Модификаторы шаблонов

Модификаторы шаблонов используются в нескольких языках, в частности, в Perl. Они позволяют изменить работу парсера. Например, модификатор `i` заставляет парсер игнорировать регистры символов.

Регулярные выражения в Perl обрамляются одним и тем же символом в начале и в конце. Это может быть любой символ (чаще используется `/.`), и выглядит все таким образом:

```
(?i)pattern
```

Мета-символы

Наконец, последняя часть таблицы содержит мета-символы. Это символы, имеющие специальное значение в регулярных выражениях. Так что если вы хотите использовать один из них как обычный символ, то его необходимо экранировать. Для проверки наличия скобки в тексте, используется такой шаблон:

```
\(
```

Примеры регулярных выражений

Код цвета в шестнадцатеричном формате

```
\#[a-fA-F][0-9]{3, 6}
```

Шестнадцатеричный код 0x00-0xFF

```
0x[a-fA-F0-9]{2}
```

Извлекаем вещественное число 0.0000

```
([0-9]+(\.[0-9]{2})?)
```

Извлекаем JSON атрибут "name": "1234"

```
"name":\s*"([^\"]+)
```

Извлекаем XML или HTML атрибут name="1234"

```
name="([^\"]+)
```

Извлекаем число, которое имеет некий префикс и пробелы до значения value : 1234

```
value\s*:\s*(\d+)
```

5.6 Форматирование данных

На следующей закладке вы можете определить формат данных для некоторых типов данных (см. рис. ниже).

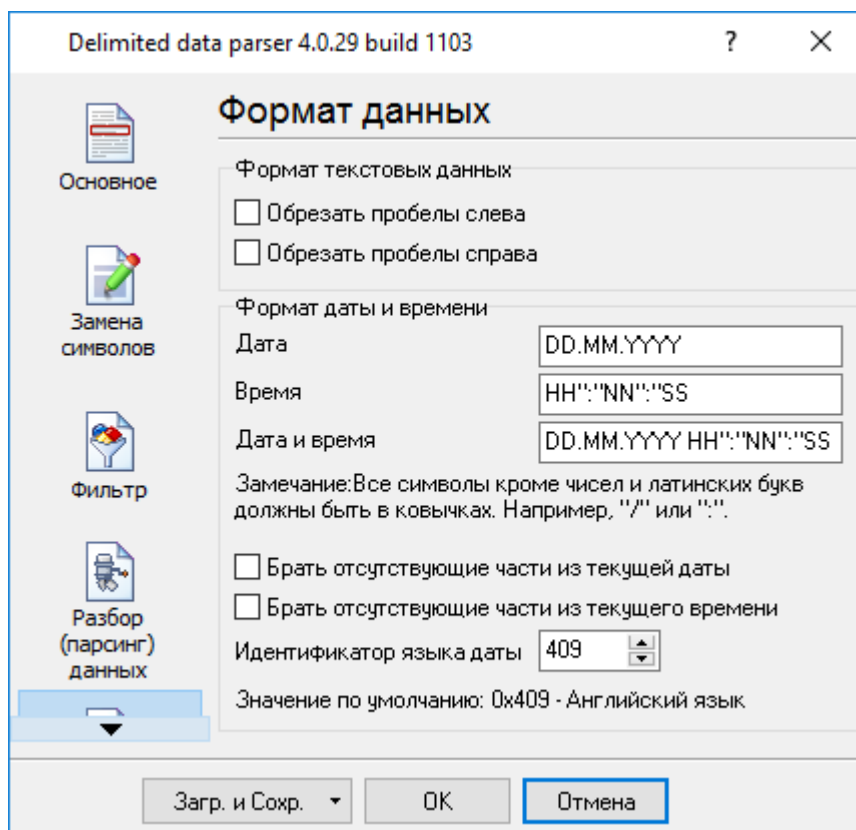


Рис. 3. Форматирование данных

Формат текстовых данных - данная группа опций позволяет обрезать пробелы с начала и/или конца переменных, имеющих тип String.

Формат даты и времени - Очень часто формат даты или времени, которое передает внешнее устройство, не совпадает с форматом, который принят для хранения в базе данных. Для того, чтобы произвести преобразование в нужный формат необходимо определить формат, в котором поступают дата и время.

Для задания формата даты и времени используется тот же формат, что используется для задания даты и времени в имени лог-файла в основной программе (например, Advanced Serial Data Logger). Замечание: символы "/" и/или ":" указываются в формате в кавычках (см. рис.).

Формат даты и времени

Пользовательский формат представления даты и времени можно задать следующим образом:

- d - день, не включает ноль (1 - 31);
- dd - день, включает ноль (01 - 31);
- ddd - день недели в текстовом формате (Пн - Вс) согласно стандарту, установленному на

данном компьютере;
dddd - день недели в полном текстовом формате (Понедельник - Воскресенье) согласно стандарту, установленному на данном компьютере;
m - месяц, не включает ноль (1 - 12);
mm - месяц, включает ноль (01 - 12);
mmm - месяц в текстовом формате (Янв - Дек) согласно стандарту, установленному на данном компьютере;
mmmm - месяц в полном текстовом формате (Январь - Декабрь) согласно стандарту, установленному на данном компьютере;
yy - год в формате двух последних разрядов (00 - 99);
yyyy - год в формате четырех последних разрядов (0000 - 9999);
h - часы, не включает ноль (0 - 23);
hh - часы, включает ноль (00 - 23);
n - минуты, не включает ноль (0 - 59);
nn - минуты, включает ноль (00 - 59);
s - секунды, не включает ноль (0 - 59);
ss - секунды, включает ноль (00-59).

Если передаваемые дата или время не содержать какой-либо части (например, год) вы можете указать брать соответствующую часть из текущей даты или времени:

- **Брать отсутствующие части из текущей даты;**
- **Брать отсутствующие части из текущего времени.**

Иногда устройства передают дату с именами месяцев (например "Jan, 10 2026") и это имя может быть на другом языке (в данном примере на английском языке), отличном от языка вашей операционной системы. В этом случае указывается идентификатор языка (language ID), на котором передается дата в поле "**Идентификатор языка даты**".

6 Проблемы?

6.1 Возможные проблемы

Модуль отсутствует в списке или помечен как "не установлен" - убедитесь, что модуль был установлен в нужную папку. Все модули должны располагаться в подпапке "Plugins" в папке с программой. Также необходимо убедиться, что модуль совместим с вашей версией логгера. Если модуль несовместим, то сообщение об этом должно появиться в логе сообщений программы в главном окне.

Нет данных для публикации (экспорта) – никаких данных не было передано для экспорта. Решение: настройте парсер, убедитесь, что в парсере объявлена одна или несколько переменных.

Ошибка при присваивании значения переменной или параметра %s [%s] – ошибка обычно возникает, если данные не соответствуют указанному формату. Например, формат даты или времени не соответствует данным.

Ошибка при формировании параметра (%s) – программа не может преобразовать переменную одного формата в другой. Это проблема возникает, когда вы пытаетесь произвести

экспорт переменной одного типа данных в столбец базы данных, который имеет другой несовместимый тип данных. Для исправления ошибки измените тип данных в парсере или в модуле экспорта данных.

Не удалось соединиться %s. (%s) – ошибка возникает, когда модуль не может подсоединиться к базе данных. Проверьте ваши настройки соединения в модуле экспорта данных.

Не удалось отсоединиться %s. (%s) – ошибка возникает, когда модуль не может отсоединиться от базы данных. Обычно, ошибка возникает при нарушении связи с базой данных (нет сети, база данных недоступна).

Если у вас возникли другие проблемы, то пишите на support@aggsoft.ru. Все решим в кратчайшие сроки.

Замечание: в тексте ошибки выше, выражение "%s" будет заменено на дополнительные данные.