MaxPatrol SIEM версия 8.0

Руководство разработчика

Дата редакции документа: 27.09.2023

Содержание

1. Об этом документе

Руководство разработчика содержит информацию, необходимую для тонкой настройки анализа данных в Positive Technologies MaxPatrol SIEM (далее также — MaxPatrol SIEM).

Руководство содержит рекомендации по созданию правил нормализации, корреляции, агрегации и обогащения событий и описание утилит MaxPatrol SIEM SDK для их отладки.

В руководстве приводятся сведения о функциях языка eXtraction and Processing, используемых при создании правил нормализации, корреляции и обогащения. В приложении дан перечень полей событий и указаны правила их заполнения.

Руководство также содержит информацию о доступных в MaxPatrol SIEM функциях сервиса REST API.

Руководство адресовано разработчикам, выполняющим настройку MaxPatrol SIEM для сбора, обработки и анализа событий источников, а также выполняющих интеграцию MaxPatrol SIEM со сторонними приложениями.

Комплект документации MaxPatrol SIEM включает в себя следующие документы:

- Этот документ.
- Руководство по внедрению — содержит информацию для внедрения продукта в инфраструктуре организации: от типовых схем развертывания до инструкций по установке, первоначальной настройке, обновлению и удалению продукта.
- Руководство администратора — содержит справочную информацию и инструкции по настройке и администрированию продукта.
- Руководство оператора безопасности — содержит сценарии использования продукта для управления информационными активами организации и событиями информационной безопасности.
- Руководство по настройке источников — содержит рекомендации по интеграции элементов IT-инфраструктуры организации с MaxPatrol SIEM для сбора событий с источников и аудита активов.
- Синтаксис языка запроса PDQL — содержит справочную информацию и примеры синтаксиса, основных функций и операторов языка PDQL, используемых при работе с MaxPatrol SIEM.
- PDQL-запросы для анализа активов — содержит информацию о стандартных запросах на языке PDQL, предназначенных для проверки конфигураций активов при работе в MaxPatrol SIEM.

В этом разделе

Условные обозначения (см. раздел 1.1)

Другие источники информации о MaxPatrol SIEM (см. раздел 1.2)

1.1. Условные обозначения

В документе приняты условные обозначения.

Таблица 1. Условные обозначения

Пример	Описание
Внимание! При выключении модуля снижается уровень защищенности сети	Предупреждения. Содержат информацию о действиях или событиях, которые могут иметь нежелательные последствия
Примечание. Вы можете создать дополнительные отчеты	Примечания. Содержат советы, описания важных частных случаев, дополнительную или справочную информацию, которая может быть полезна при работе с продуктом
► Чтобы открыть файл:	Начало инструкции выделено специальным значком
Нажмите кнопку **OK**	Названия элементов интерфейса (например, кнопок, полей, пунктов меню) выделены полужирным шрифтом
Выполните команду `Stop-Service`	Текст командной строки, примеры кода, прочие данные, которые нужно ввести с клавиатуры, выделены специальным шрифтом. Также выделены специальным шрифтом имена файлов и пути к файлам и папкам
Ctrl+Alt+Delete	Комбинация клавиш. Чтобы использовать комбинацию, клавиши нужно нажимать одновременно
<Название программы>	Переменные заключены в угловые скобки

1.2. Другие источники информации о MaxPatrol SIEM

Вы можете найти дополнительную информацию о MaxPatrol SIEM на портале технической поддержки.

Портал содержит статьи базы знаний, новости обновлений продуктов Positive Technologies, ответы на часто задаваемые вопросы пользователей. Для доступа к базе знаний и всем новостям нужно создать на портале учетную запись.

Если вы не нашли нужную информацию или решение проблемы самостоятельно, обратитесь в службу технической поддержки (см. раздел 17).

2. Этапы обработки событий

MaxPatrol SIEM выполняет непрерывный мониторинг информационной безопасности IT-инфраструктуры организации. Мониторинг осуществляется путем сбора событий с источников и отслеживания состояния активов.

Источником событий может являться любой элемент IT-инфраструктуры, для которого настроены журналирование и сбор событий. Для описания IT-инфраструктуры используется модель. Активами называются объекты модели, соответствующие элементам IT-инфраструктуры. Изменения состояний активов фиксируются на основе данных аудита IT-инфраструктуры организации.

Для обнаружения событий информационной безопасности используется метод rule-based reasoning. Специалисты по информационной безопасности заранее создают правила, описывающие признаки события. Событие информационной безопасности фиксируется, если в потоке событий от источников появляется событие (или последовательность событий), указанное в одном из правил, или фиксируется изменение состояния актива, описанное в одном из правил.

Сбор событий

Сбор событий с источников выполняется компонентом MP 10 Collector. Компонент имеет модульную структуру и в зависимости от используемых модулей может выполнять активный или пассивный сбор необработанных событий с источников или аудит активов. В зависимости от особенностей хранения событий на источнике (например, сетевая папка, журнал ОС, СУБД) используются специализированные алгоритмы сбора событий. Для описания особенностей того или иного источника для модулей создаются шаблоны настройки — профили.

Нормализация событий

Для событий, собранных с разных источников, могут отличаться формат (TXT, XML, JSON) и стандарт записи. Для анализа потока событий требуется преобразовать все события к единому виду.

Нормализация события — процедура приведения необработанного события к нормализованному виду в соответствии с заранее заданным для источника и типа события правилом нормализации.

Нормализованное событие — событие представляет собой совокупность полей, заполненных данными из необработанного события согласно правилу нормализации.

> **Примечание.** Для записи событий (нормализованных, агрегированных, корреляционных) в MaxPatrol SIEM используется внутренний стандарт, разработанный на основе стандарта Common Event Expression (его описание см. на сайте mitre.org).

Агрегация событий

Поток событий может содержать однотипные события, отличающиеся значением одного или нескольких полей (например, временем регистрации). Для сокращения количества таких событий выполняется агрегация событий.

Агрегация событий — это процесс отбора (в потоке нормализованных и корреляционных событий) событий, которые удовлетворяют условию заранее настроенного правила агрегации, и объединения их в одно агрегированное событие.

Обогащение событий

Обогащение событий — заполнение полей нормализованных, агрегированных и корреляционных событий согласно правилам обогащения. Поля заполняются данными, указанными в правиле обогащения или полученными из табличных списков.

Табличный список — двумерный массив данных, хранящийся в памяти MaxPatrol SIEM и доступный для использования в правилах корреляции и правилах обогащения.

Корреляция событий

Корреляция событий — это процесс обнаружения событий информационной безопасности путем анализа потока нормализованных событий. При обнаружении в потоке событий такой их последовательности, которая указана в условии одного из заранее настроенных правил корреляции, регистрируется корреляционное событие.

Корреляционное событие — событие информационной безопасности, представляющее собой совокупность полей, заполненных данными о возможном нарушении согласно правилам, указанным в правиле корреляции.

Инцидент — корреляционное событие, связанное с нарушением информационной безопасности. Сведения об инцидентах оперативно передаются оператору MaxPatrol SIEM.

3. Программные компоненты для обработки событий

MP 10 Collector — компонент MaxPatrol SIEM для сбора событий с источников и аудита IT-инфраструктуры предприятия. Формирует пакеты необработанных событий и отправляет их через MP 10 Core на MP SIEM Server.

MP 10 Core — сервер управления MaxPatrol SIEM, обеспечивающий доступ пользователей через веб-интерфейс и хранение информации об активах.

SIEM Server receiver — служба для приема событий. Выполняет предобработку событий и передает необработанные события службе нормализации.

SIEM Server normalizer — служба для нормализации необработанных событий источников в соответствии с указанными заранее для каждого источника и типа событий правилами нормализации. Нормализованные события передаются службе агрегации, ненормализованные — службе SIEM Server resolver.

SIEM Server correlator — служба для регистрации событий информационной безопасности в IT-инфраструктуре организации путем проверки потока нормализованных (и корреляционных) событий на соответствие указанным заранее правилам корреляции. Зарегистрированные корреляционные события передаются службе агрегации.

SIEM Server aggregator — служба для анализа потока нормализованных и корреляционных событий на соответствие указанным заранее правилам агрегации. События, удовлетворяющие правилу агрегации, объединяются в одно агрегированное событие, исходные события удаляются. Агрегированные события и события, не удовлетворяющие правилам агрегации (нормализованные, и корреляционные), передаются службе SIEM Server resolver.

SIEM Server resolver — служба для поиска связи событий с зарегистрированными в MaxPatrol SIEM активами. Для каждого нормализованного события выполняется запрос к MP 10 Core на наличие актива, параметры которого соответствуют параметрам источника события. Если актив существует, то в событии указывается UUID актива. Нормализованные и корреляционные события передаются службе обогащения, ненормализованные — службе SIEM Server storage.

SIEM Server enricher — служба для обогащения нормализованных и корреляционных событий данными в соответствии с указанными заранее правилами обогащения. События передаются службе SIEM Server router.

SIEM Server router — служба для маршрутизации и предварительной обработки событий. Нормализованные и корреляционные события передаются службе SIEM Server storage. Из потока нормализованных и корреляционных событий отбираются удовлетворяющие условиям правил корреляции и передаются службе корреляции. Зарегистрированные корреляционные события передаются службе Notifier.

SIEM Server storage — служба для работы с хранилищем событий. Все поступающие нормализованные, ненормализованные и корреляционные события сохраняются в БД.

Notifier — сервис в составе службы корреляции для передачи информации об инцидентах в MP 10 Core.

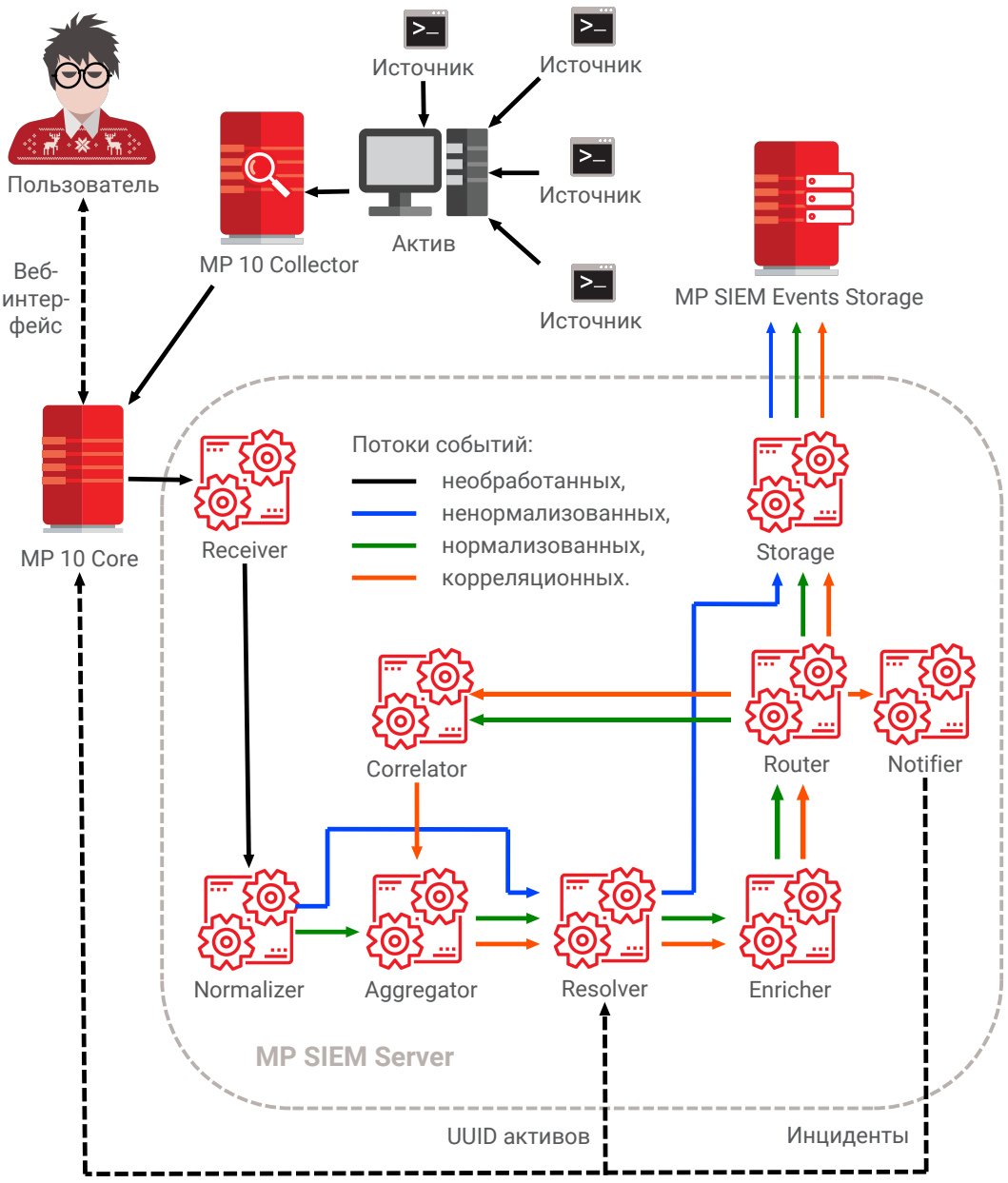

Рисунок 1. Обработка событий в MaxPatrol SIEM

4. Язык eXtraction and Processing

Язык eXtraction and Processing (далее также — XP) разработан в Positive Technologies для создания правил преобразования данных в процессе обработки событий источников в MaxPatrol SIEM. Язык XP используется при написании правил нормализации, корреляции и обогащения событий.

В разделе описаны синтаксис, основные функции и операторы языка XP, приводятся примеры использования. Некоторые функции языка XP используются только при написании правил нормализации, другие только при написании правил корреляции или обогащения событий, это будет указано дополнительно в описании функций. Полный перечень функций, используемых при написании правил нормализации, корреляции и обогащения, приведен в предметном указателе.

В этом разделе

4.1. Имена

В языке XP используются следующие имена:

- Переменные, например `$f1`. Имена всех создаваемых переменных нужно начинать со знака доллара ($). В именах можно использовать латинские буквы, цифры, точку и символ подчеркивания. Переменным можно присваивать значения любых типов данных.
- Функции, например: `number($f1)`, `regex($f1, $f2, $f3)`, `coalesce($f1, $f2, $f3,...)`. В круглых скобках указываются аргументы функции. Имя, количество аргументов (кроме `coalesce`) и приведение типов данных предопределены для каждой функции.

  Примечание. Функции с префиксом `maybe_` используются при отладке кода с помощью утилит комплекта разработчика MaxPatrol SIEM SDK.
- Поля событий, например: `action`, `event_src.ip`, `correlation_name`, `datafield1`. Перечни полей событий предопределены для каждого типа события и отличаются друг от друга. Каждое поле может принимать значения только одного типа данных.

— Ключевые слова форматной строки правил нормализации для указания типа анализируемого события, например: `EVENTLOG`, `JSON`, `TEXT`, `TABULAR`, `XML`. Ключевое слово строки с условием выполнения правила нормализации: `COND`.

Примечание. Имена ключевых слов в правиле нормализации могут начинаться с восклицательного знака (!). Это устаревший вариант синтаксиса языка XP.

— Токены форматной строки правила нормализации для анализа текстовой строки события, например: `{CSV}`, `{DATETIME}`, `{DURATION}`, `{HOSTNAME}`, `{IPV4}`, `{IPV6}`, `{KEYVALUE}`, `{LITERAL}`, `{NTUSER}`, `{NUMBER}`, `{REST}`, `{STRING}`, `{WORD}`, `{WORDDASH}`.

— Директивы и инструкции правила корреляции и обогащения, например: `query`, `event`, `rule`, `emit`, `enrichment`, `enrich`.

— Названия правил корреляции и обогащения (`Critical_login_success`, `Map_protocol_number_to_name`), названия событий и запросов, объявляемые в правилах корреляции и обогащения, названия табличных списков (`Log_names`, `Known_Malwares`). Все названия должны начинаться с прописной буквы, в них можно использовать латинские буквы, цифры и символ подчеркивания.

— Префиксы, например: `column::` — для обращения к колонке табличного списка, `filter::` — для вызова макроса или `table::` — для выполнения инлайн-запроса к табличному списку.

4.2. Математические и логические операторы

В разделе описаны математические и логические операторы языка XP. При вычислении выражений операторы применяются согласно приоритету.

Таблица 2. Приоритет математических и логических операторов

Приоритет	Оператор	Описание
1	Вызов функции	—
2	–, not	Унарный минус и логическое отрицание
3	*, div, mod	Умножение, деление и деление по модулю
4	+, –	Сложение и вычитание
5	< , <=, >, >=	Логическое сравнение
6	==, !=	Логические равенство и неравенство
7	and	Конъюнкция (логическое И)
8	or	Дизъюнкция (логическое ИЛИ)
9	=	Присваивание

В этом разделе

4.2.1. Присваивание «=»

Результатом применения оператора является присваивание переменной значения литерала, другой переменной, функции или выражения.

Формат вызова:

```
$f1 = $f2
```

Пример

```
# Необработанное событие:
# Example: 12
TEXT = 'Example: {$f2=NUMBER}'
$f1 = $f2
datafield1 = $f1
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "12",
#    "time": "2017-10-18T12:00:00Z"}
```

4.2.2. Сложение «+»

Результатом применения оператора к числовому типу данных Number является сумма операндов. Если оба операнда имеют тип данных String, результатом является конкатенация (сцепление) операндов в одну строку. Если первый операнд имеет тип данных List, результатом является список, дополненный вторым операндом.

Формат вызова:

```
$f1 + $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 + $f2
datafield2 = string($f1) + string($f2)
$f3 = [144, 72] + $f1 * $f2
datafield3 = $f3[2]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "14",
#    "datafield2": "122",
#    "datafield2": "24",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 3. Приведение типов данных оператора сложения

Первый операнд	Второй операнд	Результат
Number	Number, Null	Number
String	String, Null	String
DateTime	Number, Null	DateTime
List	Number, String, DateTime	List

4.2.3. Вычитание «–»

Результатом применения оператора к числовому типу данных Number является разность операндов. При использовании в качестве унарного оператора результатом является арифметическое отрицательное значение операнда.

Формат вызова:

```
$f1 - $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 - $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "10,
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 4. Приведение типов данных оператора вычитания

Первый операнд	Второй операнд	Результат
Number	Number, Null	Number
String	Null	String

4.2.4. Умножение «*»

Результатом применения оператора к числовому типу данных Number является произведение операндов.

Формат вызова:

```
$f1 * $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 * $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "24",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 5. Приведение типов данных оператора умножения

Первый операнд	Второй операнд	Результат
Number, Null	Number	Number
Number	Number, Null	Number

4.2.5. Деление «div»

Результатом применения оператора `div` является отношение первого операнда ко второму. Если оба операнда относятся к типу данных Number, результатом является целая часть от деления операндов. Деление на ноль не допускается.

Формат вызова:

```
div($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = div($f1, $f2)
datafield2 = div($f1, 7)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "6",
#    "datafield2": "1",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 6. Приведение типов данных оператора деления

Первый операнд	Второй операнд	Результат
Number	Number	Number

4.2.6. Деление по модулю «mod»

Результатом применения оператора `mod` к целым числам является остаток от целочисленного деления операндов. Если первый операнд меньше второго, то результатом является его значение. Деление на ноль не допускается.

Формат вызова:

```
mod($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example: 12 7 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER} {$f3=NUMBER}'
datafield1 = mod($f1, $f2)
datafield2 = mod($f1, $f3)
datafield3 = mod($f3, $f2)
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1": "5",
#    "datafield2": "0",
#    "datafield3": "2",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 7. Приведение типов данных оператора деления по модулю

Первый операнд	Второй операнд	Результат
Number	Number	Number

4.2.7. Равенство «==»

Если первый операнд равен второму, результатом применения оператора равенства является значение True; в противном случае — значение False.

Формат вызова:

```
$f1 == $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 == $f2
datafield2 = $f1 == 6 * $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "false",
#    "datafield2": "true",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 8. Приведение типов данных логического оператора равенства

Оба операнда	Результат
Number, DateTime, String, IPAddress	Bool

4.2.8. Неравенство «!=»

Если операнды не равны, результатом применения оператора является значение True; в противном случае — значение False.

Формат вызова:

```
$f1 != $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 != $f2
datafield2 = $f1 != 6 * $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 9. Приведение типов данных логического оператора неравенства

Оба операнда	Результат
Number, DateTime, String, IPAddress	Bool

4.2.9. Сравнение: «<», «>»

Результатом применения оператора «меньше» (<) является значение True, если первый операнд меньше второго, в противном случае — значение False.

Результатом применения оператора «больше» (>) является значение True, если первый операнд больше второго, в противном случае — значение False.

Формат вызова:

```
$f1 < $f2
$f1 > $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 > $f2
datafield2 = $f1 < $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 10. Приведение типов данных операторов сравнения

Оба операнда	Результат
Number, DateTime	Bool

4.2.10. Сравнение: «>=», «<=»

Результатом применения оператора «меньше или равно» (<=) является значение True, если первый операнд меньше или равен второму, в противном случае — значение False.

Результатом применения оператора «больше или равно» (>=) является значение True, если первый операнд больше или равен второму, в противном случае — значение False.

Формат вызова:

```
$f1 >= $f2
$f1 <= $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 >= $f2
datafield2 = $f1 >= 6 * $f2
datafield3 = $f1 >= 7 * $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "true",
#    "datafield3": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 11. Приведение типов данных операторов сравнения

Оба операнда	Результат
Number, DateTime	Bool

4.2.11. Конъюнкция: «and», «&&»

Результатом применения логического оператора И является значение True, если оба операнда имеют значение True, в противном случае — значение False. Если один из операндов не задан (null), результатом применения оператора является значение False.

Формат вызова:

```
$f1 and $f2
```

или

```
$f1 && $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
datafield1 = $f1 > 0 and $f1 < $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 12. Приведение типов данных логического оператора И

Первый операнд	Второй операнд	Результат
Bool	Bool	Bool
Bool	Null	Bool

4.2.12. Дизъюнкция: «or», «||»

Результатом применения логического оператора ИЛИ является значение False, если оба операнда имеют значение False, в противном случае — значение True. Если один из операндов не задан, результатом применения оператора является значение второго операнда.

Формат вызова:

```
$f1 or $f2
```

или

```
$f1 || $f2
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2= NUMBER}'
datafield1 = $f1 > 0 or $f1 < $f2
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 13. Приведение типов данных логического оператора ИЛИ

Первый операнд	Второй операнд	Результат
Bool	Bool	Bool
Bool	Null	Bool

4.2.13. Отрицание «not»

Результатом применения оператора `not` является смена логического значения операнда на противоположное.

Формат вызова:

```
not $f1
```

Пример

```
# Необработанное событие:
# Example: 12 2
TEXT = 'Example: {$f1=NUMBER} {$f2=NUMBER}'
$f3 = $f1 > $f2
datafield1 = $f3
datafield2 = not $f3
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 14. Приведение типов данных логического оператора отрицания

Операнд	Результат
Bool	Bool

4.3. Конструкции языка

В разделе описаны конструкции для написания выражений в языке XP.

В этом разделе

Символ комментария «#» (см. раздел 4.3.1)

Объявление строки (см. раздел 4.3.2)

Объединение в список «[..., ...]» (см. раздел 4.3.3)

Условие «if» (см. раздел 4.3.4)

Множественный выбор «switch» (см. раздел 4.3.5)

4.3.1. Символ комментария «#»

Символ предназначен для объявления однострочного комментария. Все последующие символы в строке игнорируются при компиляции кода.

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
# Заполните поля
datafield1 = 'значениями' # из событий
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "значениями",
#    "time": "2017-10-18T12:00:00Z"}
```

4.3.2. Объявление строки

Результатом применения оператора " " (' ') является переменная с типом данных String. Аргументом оператора может быть только набор символов.

Формат вызова:

```
$string = "..."
```

или

```
$string = '...'
```

Пример

```
# Необработанное событие:
# Example: Click
TEXT = 'Example: {$f1=STRING}'
$f2 = " 10 "
$f3 = 'times'
datafield1 = $f1 + $f2 + $f3
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Click 10 times",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 15. Приведение типов данных при объявлении строки

Аргумент	Результат
—	String

4.3.3. Объединение в список «[..., ...]»

Результатом применения оператора [ ] является объединение аргументов в список. Отсчет элементов списка ведется от 0.

Формат вызова для формирования списка:

```
$list1 = [$f1, $f2, ...]
$list2 = [[$f11, $f12, ...], $f2, ...]
```

Формат вызова для обращения к элементу списка:

```
$f1 = $list1[0]
$f12 = $list2[0][1]
```

Пример

```
# Необработанное событие:
# Example: Click 10 times
TEXT = 'Example: {$f1=STRING} {$f2=NUMBER} {$f3=STRING}'
$list1 = [$f1, $f2, $f3]
datafield1 = $list1[0]
datafield2 = $list1[1]
datafield3 = $list1[2]
$list2 = [[$f1, $f2], $f3]
datafield4 = $list2[0][0]
datafield5 = $list2[0][1]
datafield6 = $list2[1]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Click",
#    "datafield2": "10",
#    "datafield3": "times",
#    "datafield4": "Click",
#    "datafield5": "10",
#    "datafield6": "times",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 16. Приведение типов данных при объединении в список

Аргумент	Результат
Любой	List

4.3.4. Условие «if»

Результатом применения условного оператора `if` является выполнение той или иной последовательности операторов, в зависимости от того, какое из условий возвращает значение True. Если первое условие возвращает значение True, выполняется первый блок операторов, если второе условие возвращает значение True — второй блок операторов, и так далее. Если указан альтернативный блок операторов, то он выполняется, если все условия возвращают значения False.

Формат вызова:

```
if <Условие 1> then
    <Блок операторов 1>
elif <Условие 2> then
    <Блок операторов 2>
...
else
    <Альтернативный блок операторов>
endif
```

Пример

```
# Необработанное событие:
# Example: Click 10 times
TEXT = 'Example: {STRING} {$f1=NUMBER} {STRING}'
if $f1 == 11 then
     $f2 = 10
elif $f1 == 10 then
     $f2 = 9
elif $f1 == 9 then
     $f2 = 8
else
     $f2 = 0
endif
datafield1 = string($f2) + " times left"
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "9 times left",
#    "time": "2017-10-18T12:00:00Z"}
```

4.3.5. Множественный выбор «switch»

Результатом применения оператора множественного выбора `switch` является выполнение той или иной последовательности операторов, в зависимости от значения операнда. Если операнд принимает первое значение, выполняется первый блок операторов, если операнд принимает второе значение — второй блок операторов, и так далее. Количество указанных значений операнда не ограничено.

Формат вызова:

```
switch $f1
    case <Значение 1>
        <Блок операторов 1>
    case <Значение 2>
        <Блок операторов 2>
    case <Значение 3>
        <Блок операторов 3>
    ...
endswitch
```

Пример

```
# Необработанное событие:
# Example: Click 8 times
TEXT = 'Example: {STRING} {$f1=NUMBER} {STRING}'
switch $f1
     case 10
          $f2 = 9
     case 9
          $f2 = 8
     case 8
          $f2 = 7
endswitch
datafield1 = string($f2) + ' times left'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "7 times left",
#    "time": "2017-10-18T12:00:00Z"}
```

Упрощенный формат вызова оператора

Результатом применения оператора множественного выбора является присваивание переменной того или иного значения, в зависимости от значения операнда. Если операнд принимает первое значение, переменной присваивается одно значение, если операнд принимает второе значение — другое, и так далее. Количество указанных значений операнда не ограничено.

Формат вызова:

```
$f2 = switch $f1
    case <Значение $f1 1> <Значение $f2 1>
    case <Значение $f1 2> <Значение $f2 2>
    case <Значение $f1 3> <Значение $f2 3>
    ...
endswitch
```

Пример

```
# Необработанное событие:
# Example: Click 8 times
TEXT = 'Example: {STRING} {$f1=NUMBER} {STRING}'
$f2 = switch $f1
     case 10 9
     case 9 8
     case 8 7
endswitch
datafield1 = string($f2) + ' times left'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "7 times left",
#    "time": "2017-10-18T12:00:00Z"}
```

4.4. Функции работы с временем

В разделе описаны функции языка XP для преобразования и анализа данных, содержащих время и дату.

В этом разделе

4.4.1. datetime (maybe_datetime)

Внимание! Функция может использоваться только при создании правил нормализации.

Функция `datetime` используется для преобразования значения аргумента с типом данных String в тип данных DateTime.

Формат вызова:

```
datetime($f1)
```

Пример

```
# Необработанное событие:
# Example: 1990-10-31T23:13:04+03:00
TEXT = 'Example: {$f1=STRING}'
$f2 = datetime($f1)
datafield1 = $f2
datafield2 = year($f2)
datafield3 = month($f2)
datafield4 = day($f2)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "1990-10-31T23:13:04+03:00",
#    "datafield2": "1990",
#    "datafield3": "10",
#    "datafield4": "31",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 17. Приведение типов данных функции datetime

Аргумент	Результат
String	DateTime
DateTime	DateTime

4.4.2. datetime_to_epoch, datetime_to_epoch_ms

Функции `datetime_to_epoch` и `datetime_to_epoch_ms` используются для преобразования формата времени. Аргументом функций является время с типом данных DateTime. Функции возвращают в Unix-время, функция `datetime_to_epoch` — в секундах, функция `datetime_to_epoch_ms` — в миллисекундах. Отсчет Unix-времени начинается с 00:00:00 UTC 1 января 1970 года.

Формат вызова:

```
datetime_to_epoch($f1)
datetime_to_epoch_ms($f1)
```

Пример

```
# Необработанное событие:
# Example: 2015-01-25T01:06:40
TEXT = 'Example: {$f1=DATETIME}'
datafield1 = $f1
datafield2 = datetime_to_epoch($f1)
datafield3 = datetime_to_epoch_ms($f1)
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1": "2015-01-25T01:06:40Z",
#    "datafield2": "1422148000",
#    "datafield3": "1422148000000",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 18. Приведение типов данных функций datetime_to_epoch, datetime_to_epoch_ms

Аргумент	Результат
DateTime	Number

4.4.3. datetime_to_win_ticks

Функция `datetime_to_win_ticks` используется для преобразования формата времени. Аргументом функции является время с типом данных DateTime. Функция возвращает время в формате filetime с типом данных Number. Время filetime указывается в сотнях наносекунд (10^{-7} секунды) начиная с 00:00:00 UTC 1 января 1601 года.

Формат вызова:

```
datetime_to_win_ticks($f1)
```

Пример

```
# Необработанное событие:
# Example: 2015-01-25T01:06:40
TEXT = 'Example: {$f1=DATETIME}'
datafield1 = $f1
datafield2 = datetime_to_win_ticks($f1)
datafield3 = div(datetime_to_win_ticks($f1), 10000000)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "2015-01-25T01:06:40Z",
#    "datafield2": "130666216000000000",
#    "datafield3": "13066621600",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 19. Приведение типов данных функций datetime_to_win_ticks

Аргумент	Результат
DateTime	Number

4.4.4. duration (maybe_duration)

Функция `duration` используется для преобразования значения аргумента с типом данных String, содержащего интервал времени в формате XXXd XXh:XXm:XXs, в число секунд. Если преобразование не удалось, функция возвращает null.

Формат вызова:

```
duration($f1)
```

Пример

```
# Необработанное событие:
# Example: 0d 1h:2m:15s
TEXT = 'Example: {datafield1=REST}'
$f = duration(datafield1)
datafield2 = $f
datafield3 = div($f,60*60)
datafield4 = div(mod($f,3600),60)
datafield5 = mod(mod($f,3600),60)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "0d 1h:2m:15s",
#    "datafield2": "3735",
#    "datafield3": "1",
#    "datafield4": "2",
#    "datafield5": "15",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 20. Приведение типов данных функции duration

Аргумент	Результат
String	Number
Number	Number

4.4.5. epoch_to_datetime, epoch_ms_to_datetime

Функции `epoch_to_datetime` и `epoch_ms_to_datetime` используются для преобразования формата времени. Аргументом функций является Unix-время. Нужно указать промежуток времени начиная с 00:00:00 UTC 1 января 1970 года, для функции `epoch_to_datetime` — в секундах, для функции `epoch_ms_to_datetime` — в миллисекундах. Функции возвращают значение времени с типом данных DateTime. При некорректном аргументе функции возвращают null.

Формат вызова:

```
epoch_to_datetime($f1)
epoch_ms_to_datetime($f1)
```

Пример

```
# Необработанное событие:
# Example: 1422148000
TEXT = 'Example: {$f1=NUMBER}'
datafield1 = epoch_to_datetime($f1)
datafield2 = epoch_ms_to_datetime($f1*1000)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"2015-01-25T01:06:40Z",
#    "datafield2":"2015-01-25T01:06:40Z",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 21. Приведение типов данных функций epoch_to_datetime, epoch_ms_to_datetime

Аргумент	Результат
Number	DateTime

4.4.6. hour, minute, second, timezone

Функции `hour`, `minute`, `second` и `timezone` используются для получения из даты значения часов, минут, секунд и отклонения от UTC. В аргументе функций нужно указать дату с типом данных DateTime. Функции возвращают значения в секундах.

Формат вызова:

```
hour($f1)
minute($f1)
second($f1)
timezone($f1)
```

Пример

```
# Необработанное событие:
# Example: 1990-10-31T23:13:04+03:00
TEXT = 'Example: {$f1=DATETIME}'
datafield1 = hour($f1)
datafield2 = minute($f1)
datafield3 = second($f1)
datafield4 = timezone($f1)
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1":"20",
#    "datafield2":"13",
#    "datafield3":"4",
#    "datafield4":"10800",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 22. Приведение типов данных функций hour, minute, second, timezone

Аргумент	Результат
DateTime	Number

4.4.7. win_ticks_to_datetime

Функция `win_ticks_to_datetime` используется для преобразования формата времени. Аргументом функций является время в формате filetime. Время filetime указывается в сотнях наносекунд (10^{-7} секунды) начиная с 00:00:00 UTC 1 января 1601 года. Время ранее 1 января 1970 года считается невалидным. Функции возвращают время с типом данных DateTime.

Формат вызова:

```
win_ticks_to_datetime($f1)
```

Пример

```
# Необработанное событие:
# Example: 130666216000000000
TEXT = 'Example: {$f1=NUMBER}'
datafield1 = win_ticks_to_datetime($f1)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "2015-01-25T01:06:40Z"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 23. Приведение типов данных функций win_ticks_to_datetime

Аргумент	Результат
Number	DateTime

4.4.8. year, month, day

Функции `year`, `month` и `day` используются для получения из даты значения года, месяца и числа (в часовом поясе UTC). В аргументе функций нужно указать дату с типом данных DateTime.

Формат вызова:

```
year($f1)
month($f1)
day($f1)
```

Пример

```
# Необработанное событие:
# Example: 1990-10-31T02:13:04+03:00
TEXT = 'Example: {$f=DATETIME}'
datafield1 = year($f)
datafield2 = month($f)
datafield3 = day($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "1990",
#    "datafield2": "10",
#    "datafield3": "30",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 24. Приведение типов данных функций year, month, day

Аргумент	Результат
DateTime	Number

4.5. Функции преобразования типов данных

В разделе описаны функции языка XP для преобразования типов данных.

В этом разделе

number16 (maybe_number16) (см. раздел 4.5.10)

string (см. раздел 4.5.11)

4.5.1. bool

Функция `bool` используется для преобразования значения аргумента в логическое значение True или False.

Формат вызова:

```
bool($f1)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
datafield1 = bool(tRuE)
datafield2 = bool(12)
datafield3 = bool(Hello)
datafield4 = bool(0)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "true",
#    "datafield3": "false",
#    "datafield4": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 25. Приведение типов данных функции bool

Тип данных аргумента	Значение аргумента	Результат
Bool	`True`	True
Bool	`False`	False
String	Любое, кроме: `0`, `false`, `False`, пустой строки (`""`)	True
String	`0`, `false`, `False`, пустая строка	False
Number	`1`, любое число	True
Number	`0`	False
Null	`null`	False

4.5.2. buffer_from_base64

Функция `buffer_from_base64` используется для преобразования строки (данные типа String) в массив байтов типа Buffer. Данные типа Buffer не отображаются в полях нормализованного сообщения, но могут быть преобразованы в тип String с помощью функции decode (см. раздел 4.5.3).

Формат вызова:

```
buffer_from_base64($f)
```

Таблица 26. Приведение типов данных функции buffer_from_base64

Аргумент	Результат
String	Buffer

4.5.3. decode

Функция `decode` используется для преобразования данных типа Buffer в строку (данные типа String). Имеет один аргумент, в котором можно указать кодировку строки — UTF-8 или UCS2-LE.

Формат вызова:

```
decode($f, "UTF-8")
```

Пример

```
# Необработанное событие:
# Example: aGVsbG8gd29ybGQ=
TEXT = 'Example: {$f=BASE64}'
time = "2018-04-27T10:52:47Z"
datafield1 = decode(buffer_from_base64($f), "UTF-8")
# Нормализованное событие: {
#    "time": "2018-04-27T10:52:47Z",
#    "datafield1": "hello world"}
```

Таблица 27. Приведение типов данных функции decode

Аргумент	Результат
Buffer	String

4.5.4. ipv4 (maybe_ipv4)

Функция `ipv4` используется для преобразования значения аргумента, содержащего IP-адрес стандарта IPv4 с типом данных String, в IP-адрес с типом данных IPAddress. Если IP-адрес распознать не удалось, функция возвращает null.

Формат вызова:

```
ipv4($f)
```

Пример

```
# Необработанное событие:
# Example: 192.168.0.1
TEXT = 'Example: {$f=STRING}'
if ipv4($f) != null then
    src.host = ipv4($f)
elif ipv6($f) != null then
    src.host = ipv6($f)
endif
time = "2021-10-18T12:00:00Z"
# Нормализованное событие: {
#    "src.host": "192.168.0.1"
#    "time": "2021-10-18T12:00:00Z"}
```

Таблица 28. Приведение типов данных функции ipv4

Аргумент	Результат
String, IPAddress	IPAddress

4.5.5. ipv6 (maybe_ipv6)

Функция `ipv6` используется для преобразования значения аргумента, содержащего IP-адрес стандарта IPv6 с типом данных String, в IP-адрес с типом данных IPAddress. Если IP-адрес распознать не удалось, функция возвращает null.

В аргументе функции поддерживаются IP-адреса в форматах X:X:X:X:X:X:X.X, X:X:X:X:X:X:D.D.D.D и X:X:X:X:X:X:X.X%S, где X — шестнадцатеричные значения для 16-битовых частей адреса, D — десятичные значения для 8-битовых частей адреса (D.D.D.D — адрес стандарта IPv4), S — четырехзначное десятичное число. Для замены одного или нескольких нулей в расположенных рядом 16-битовых частях адреса допускается использование двух двоеточий (::).

Формат вызова:

```
ipv6($f)
```

Пример

```
# Необработанное событие:
# Example: 211:DB8:0:0:8:800:200C:47A 211:DB8:0:0:8:800:192.168.0.1
211:DB8:0:0:8:800:200C:47A%5141
TEXT = 'Example: {$f1=STRING} {$f2=STRING} {$f3=STRING}'
event_src.host = ipv6($f1)
src.host = ipv6($f2)
```

```
dst.host = ipv6($f3)
time = "2021-10-18T12:00:00Z"
# Нормализованное событие: {
#    "event_src.host": "211:DB8:0:0:8:800:200C:47A",
#    "src.host": "211:DB8:0:0:8:800:192.168.0.1",
#    "dst.host": "211:DB8:0:0:8:800:200C:47A",
#    "time": "2021-10-18T12:00:00Z"}
```

Таблица 29. Приведение типов данных функции ipv6

Аргумент	Результат
String, IPAddress	IPAddress

4.5.6. join

Функция `join` используется для объединения в строку элементов списка, указанного в первом аргументе. Во втором аргументе можно указать строку или число, которое будут добавлено в строку в качестве разделителя между элементами списка.

Формат вызова:

```
join($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f1 = ["AC", "DC"]
datafield1 = join($f1)
datafield2 = join($f1, "|")
datafield3 = $f1[0] + "|" + $f1[1]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "ACDC",
#    "datafield2": "AC|DC",
#    "datafield3": "AC|DC",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 30. Приведение типов данных функции join

Первый аргумент	Второй аргумент	Результат
List	String, Number, Null	String
Null	String, Number	String

4.5.7. macaddr (maybe_macaddr)

Внимание! Функция может использоваться только при создании правил нормализации.

Функция `macaddr` используется для преобразования значения аргумента, содержащего MAC-адрес с типом данных String, в тип данных MACAddress. MAC-адрес может вводится в следующих форматах: `aa:bb:cc:dd:ee:ff`, `aa-bb-cc-dd-ee-ff`, `aabb.ccdd.eeff`.

Формат вызова:

```
macaddr($f1)
```

Пример

```
# Необработанное событие:
# Example: a1-f6-33-4d-21-56
TEXT = 'Example: {$f=STRING?}'
datafield1 = macaddr($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "A1:F6:33:4D:21:56",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 31. Приведение типов данных функции macaddr

Аргумент	Результат
String, MACAddress	MACAddress

4.5.8. number (maybe_number)

Функция `number` используется для преобразования значение аргумента, содержащего число в десятичной системе счисления с типом данных String, в число с типом данных Number. Преобразуются все цифры, расположенные в строке слева от первого нечислового символа. Если число распознать не удалось, функция возвращает null.

Формат вызова:

```
number($f1)
```

Пример

```
# Необработанное событие:
# Example: -245
TEXT = 'Example: {$f=REST}'
datafield1 = number($f)
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1": "-245"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 32. Приведение типов данных функции number

Аргумент	Результат
String, Number	Number

4.5.9. number8 (maybe_number8)

Функция `number8` используется для преобразования значения аргумента, содержащего число в восьмеричной системе счисления с типом данных String, в число с типом данных Number. Преобразуются все цифры от 0 до 7, расположенные в строке слева от первого символа, не относящегося к указанным. Если число распознать не удалось, функция возвращает null.

Формат вызова:

```
number8($f1)
```

Пример

```
# Необработанное событие:
# Example: -245
TEXT = 'Example: {$f=REST}'
datafield1 = number8($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "-365",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 33. Приведение типов данных функции number8

Аргумент	Результат
String, Number	Number

4.5.10. number16 (maybe_number16)

Функция `number16` используется для преобразования значения аргумента, содержащего число в шестнадцатеричной системе счисления с типом данных String, в число с типом данных Number. В число преобразуется набор цифр от 0 до 9 и символов от a до f, расположенных в строке слева от первого символа, не относящегося к указанным. Если число распознать не удалось, функция возвращает null.

Формат вызова:

```
number16($f1)
```

Пример

```
# Необработанное событие:
# Example: -245e
TEXT = 'Example: {$f=REST}'
datafield1 = number16($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "-9310",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 34. Приведение типов данных функции number16

Аргумент	Результат
String, Number	Number

4.5.11. string

Функция `string` используется для преобразования значение аргумента в тип данных String.

Формат вызова:

```
string($f1)
```

Пример

```
# Необработанное событие:
# Example: 192.168.0.1
TEXT = 'Example: {$f1=IPV4}'
datafield1 = string($f1) + " — IP-адрес"
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "192.168.0.1 — IP-адрес"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 35. Приведение типов данных функции string

Аргумент	Результат
Bool, Number, String, DateTime	String

4.6. Функции для работы со строками

В разделе описаны функции языка XP для преобразования данных с типом String.

В этом разделе

4.6.1. csv

Внимание! Функция может использоваться только при создании правил нормализации.

Функция `csv` используется для разбора строки с CSV-разметкой на элементы. В первом аргументе нужно указать строку для разбора. Во втором аргументе — символ, разделяющий элементы строки, в третьем аргументе — символ экранирования, стоящий перед каждым элементом строки и после него. Если элементы строки не экранированы, в третьем аргументе нужно указать любой символ, не встречающийся в строке.

Функция возвращает список элементов строки. Если первым аргументом функции является список, то значения остальных аргументов при разборе игнорируются, функция возвращает этот же список.

Формат вызова:

```
csv($f1, $f2, $f3)
```

Пример

```
# Необработанное событие:
# Example: /login=admin/, /password=supercleveradmin/
TEXT = 'Example: {$f1=REST}'
$f2 = ','
$f3 = '/'
datafield1 = csv($f1, $f2, $f3)[0]
datafield2 = csv($f1, $f2, $f3)[1]
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1":"login=admin",
#    "datafield2":"password=supercleveradmin",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 36. Приведение типов данных функции csv

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	String	String	List
List	String	String	List

4.6.2. find_substr

Функция `find_substr` используется для поиска сочетания символов в строке. В первом аргументе указывается строка для поиска, во втором — искомое сочетание символов. Функция возвращает номер байта в строке, с которого начинается искомое сочетание. Отсчет байтов начинается от нуля. Если строка не содержит указанного сочетания символов, функция возвращает null.

Формат вызова:

```
find_substr($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example: 7896453223, abcdefghijk
TEXT = 'Example: {$f1=WORD}, {$f2=STRING}'
datafield1 = find_substr($f1, "64")
datafield2 = find_substr($f2, "de")
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "3",
#    "datafield2": "3",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 37. Приведение типов данных функции find_substr

Первый аргумент	Второй аргумент	Результат
String	String	Number, Null

4.6.3. keyvalue

Внимание! Функция может использоваться только при создании правил нормализации.

Функция `keyvalue` используется для анализа строки и создания ассоциативного массива пар «ключ — значение».

Функция имеет четыре аргумента: в первом нужно указать строку для анализа, во втором — внешний разделитель между парами, в третьем — внутренний разделитель в паре, в четвертом — символ экранирования значения, стоящий перед каждым значением в паре и после него. Функция возвращает ассоциативный массив, каждый элемент которого содержит пару «ключ — значение».

> **Примечание.** Для ввод одинарных кавычек используйте обратную косую черту (\), для ввода двойных кавычек — две обратные косые черты (\\).

Формат вызова:

```
$result_list = keyvalue($f1, "<Разделитель между парами>","<Разделитель в паре>",
"<Экранирование значения>")
```

Формат обращения к элементам списка:

```
<Значение> = $result_list["<Ключ>"]
```

Пример

```
# Необработанное событие:
# Example: login='admin'; password='supercleveradmin'
TEXT = 'Example: {$f=REST}'
$result_list = keyvalue($f, ";", "=", "\'")
datafield1 = $result_list["login"]
datafield2 = $result_list["password"]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "admin",
#    "datafield2": "supercleveradmin",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 38. Приведение типов данных функции keyvalue

Первый аргумент	Второй аргумент	Третий аргумент	Четвертый аргумент	Результат
String	String	String	String	KeyValue

4.6.4. length

Функция `length` для аргумента, имеющего тип данных String, возвращает размер строки в байтах, для аргумента, имеющего тип данных List — количество элементов в списке.

Формат вызова:

```
length($f1)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f1_string = "12 Angry Men"
$f1_list = [12, " Angry", " Men"]
$f2_string = "2128506"
$f2_list = [2, 12, "85", 06]
datafield1 = length($f1_string)
datafield2 = length($f1_list)
datafield3 = length($f2_string)
datafield4 = length($f2_list)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "12",
#    "datafield2": "3",
#    "datafield3": "7",
#    "datafield4": "4",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 39. Приведение типов данных функции length

Аргумент	Результат
List, String	Number
Null	Null

4.6.5. lower

Функция `lower` используется для преобразования в строке, указанной в аргументе, всех прописных буквы в строчные.

Формат вызова:

```
lower($f1)
```

Пример

```
# Необработанное событие:
# Example: Online_SERVICE
TEXT = 'Example: {$f=STRING}'
datafield1 = lower($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "online_service",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 40. Приведение типов данных функции lower

Аргумент	Результат
String	String

4.6.6. match

Внимание! Функция может использоваться только при создании правил корреляции и обогащения.

Функция `match` используется для сравнения строки с заданным шаблоном. В первом аргументе указывается строка для поиска. Во втором аргументе указывается шаблон для поиска. В шаблоне можно использовать замещающие символы: ? — один символ, * — несколько символов. Функция возвращает True, если строка соответствует шаблону, в противном случае — False. Если какой-либо из аргументов не задан, функция возвращает null.

Формат вызова:

```
match($f1, $f2)
```

Пример

```
# Нормализованное событие: {
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    init {
        $f1 = "rh24!keykll;k5v7ke6!"
        $f2 = "*k?y*"
    }
    on Event{
        $datafield1 = match($f1, $f2)
```

```
        }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"true",
#    ...}
```

Таблица 41. Приведение типов данных функции match

Первый аргумент	Второй аргумент	Результат
String	String	Bool
String	Null	Null
Null	String	Null

4.6.7. regex

Внимание! Функция может использоваться только при создании правил корреляции и обогащения.

Функция `regex` используется для поиска в строке по заданному шаблону. В первом аргументе функции указывается строка для поиска. Во втором аргументе — шаблон для поиска. В шаблоне нужно использовать синтаксис регулярных выражений библиотеки RE2. Описание библиотеки см. на сайте github.com в разделе компании Google. В третьем аргументе указывается номер группы символов шаблона: 0 — возвращается результат для всего шаблона; 1 — возвращается результат для первой группы символов; 2 — возвращается результат для второй группы символов, и так далее. Для объединения символов шаблона в группу используются круглые скобки.

Функция возвращает строку, содержащую первую найденную последовательность символов, удовлетворяющую шаблону. Если указан отличный от нуля номер группы символов шаблона, функция возвращает часть строки, соответствующую этой группе в шаблоне. Если какой-либо из аргументов не задан или удовлетворяющая шаблону последовательность символов не найдена, функция возвращает null.

Формат вызова:

```
regex($f1, $f2, $f3)
```

Пример

```
# Нормализованное событие: {
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
     init {
          $f1 = "Outlook 14.0; Windows 8.1; Office 14.0; Windows Vista"
          $f2 = "Windows ([0-9\\.]+)"
     }
     on Event{
          $datafield1 = regex($f1, $f2, 0)
          $datafield2 = regex($f1, $f2, 1)
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"Windows 8.1",
#    "datafield2":"8.1",
#    ...}
```

Таблица 42. Приведение типов данных функции regex

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	String	Number	String

4.6.8. replace

Функция `replace` используется для замены в строке сочетания символов, удовлетворяющих шаблону, другим сочетанием символов.

Функция имеет четыре аргумента: в первом нужно указать строку для анализа; во втором — сочетание символов, которое будет подставлено в строку; в третьем — шаблон для поиска сочетания символов, которое будет удалено из строки (в шаблоне можно использовать регулярные выражения, указанные в таблице ниже); в четвертом можно указать максимальное количество вхождений, которое может быть заменено (отсчет выполняется слева направо; если значение аргумента не указано или указан ноль, количество вхождений не ограничено).

Примечание. В шаблоне для экранирования символов `\`, `*`, `?`, `[`, `!` нужно использовать `\`.

Функция возвращает копию строки, указанной в первом аргументе, в которой сочетания символов, удовлетворяющие шаблону, указанному в третьем аргументе, заменены на сочетание символов, указанное во втором аргументе.

Таблица 43. Регулярные выражения для составления шаблона

Выражение	Назначение
`?`	Замена одного любого символа. Например, для регулярного выражения `<Символ 1>?<Символ 2>` в строке будут заменены три символа, два указанных и один любой символ между ними
`*`	Замена любого количества любых символов подряд (или ни одного символа). Например, для регулярного выражения `<Символ>*` в строке будут заменены указанный символы и все последующие; для выражения `*<Символ>` — указанный символ и все перед ним; для выражения `<Символ 1>*<Символ 2>` — указанные символы и любое количество любых символов между ними
`[abcd]`	Замена любого из перечисленных символов
`[a-d]` `[0-4]`	Замена любого символа из диапазона
`[!abcd]`	Замена любого символа, кроме перечисленных
`[!a-d]` `[!0-4]`	Замена любого символа, кроме символов из диапазона

Формат вызова:

```
replace($f1, $f2, $f3, $f5)
```

Пример

```
# Необработанное событие:
# Example: a()bb(c)ddd(ee)ffff(gggg)
TEXT = 'Example: {$f1=STRING}'
datafield1 = replace($f1, ".", "?")
datafield2 = replace($f1, ".", "(?)")
datafield3 = replace($f1, ".", "(?")
datafield4 = replace($f1, ".", "(*)")
datafield5 = replace($f1, ".", "(*?)")
datafield6 = replace($f1, ".", "*)")
datafield7 = replace($f1, ".", "(*")
datafield8 = replace($f1, ".", "(*?")
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": ".......................",
#    "datafield2": "a()bb.ddd(ee)ffff(gggg)",
#    "datafield3": "a.bb.)ddd.e)ffff.ggg)",
#    "datafield4": "a.bb.ddd.ffff.",
#    "datafield5": "a.ddd.ffff.",
#    "datafield6": "....",
#    "datafield7": "a.",
#    "datafield8": "a.bb.)ddd.e)ffff.ggg)",
#    "time": "2017-10-18T12:00:00Z"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 44. Приведение типов данных функции replace

Первый аргумент	Второй аргумент	Третий аргумент	Четвертый аргумент	Результат
String	String	String	Number	String

4.6.9. rot13

Функция `rot13` используется для смещения символов латинского алфавита в строке (данные типа String) на 13 позиций. Другие символы (пробелы, специальные символы, кириллица) не попадают под действие функции и не меняются.

Формат вызова:

```
rot13($f)
```

Пример

```
# Необработанное событие:
# Example: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
TEXT = 'Example: {$f=STRING}'
datafield1 = rot13($f)
time = "2018-04-27T10:52:47Z"
# Нормализованное событие: {
#    "datafield1": "NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm",
#    "time": "2018-04-27T10:52:47Z"}
```

Таблица 45. Приведение типов данных функции rot13

Аргумент	Результат
String	String

4.6.10. strip

Функция `strip` используется для удаления из значения первого аргумента, с типом данных String, префикса, указанного во втором аргументе, и (или) постфикса, указанного в третьем аргументе.

Формат вызова:

```
strip($f1, $f2, $f3)
```

Пример

```
# Необработанное событие:
# Example: <<<<Счастье есть>>>>
TEXT = 'Example: {$f1=REST?}'
$f2 = "<<<<"
$f3 = ">>>>"
datafield1 = strip($f1, $f2, $f3)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Счастье есть",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 46. Приведение типов данных функции strip

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	String	String	String
String	String	Null	String
String	Null	String	String

Первый аргумент	Второй аргумент	Третий аргумент	Результат
Null	String, Null	String, Null	Null

4.6.11. substr

Функция `substr` возвращает часть строки, указанной в первом аргументе, начиная с байта, указанного во втором аргументе, и длиной в число байт, указанное в третьем аргументе. Если второй аргумент отрицательный, отсчет байтов выполняется с конца строки.

Формат вызова:

```
substr($f1, $f2, $f3)
```

Пример

```
# Необработанное событие:
# Example: 7896453223, abcdefghijk
TEXT = 'Example: {$f1=WORD}, {$f2=STRING}'
$f3 = 4
$f4 = 6
datafield1 = substr($f1, $f3, $f4)
datafield2 = substr($f1, -$f3, $f4)
datafield3 = substr($f2, $f3, $f4)
datafield4 = substr($f2, -$f3, $f4)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "453223",
#    "datafield2": "3223",
#    "datafield3": "efghij",
#    "datafield4": "hijk",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 47. Приведение типов данных функции substr

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	Number	Number	String

4.6.12. upper

Функция `upper` используется для преобразования в строке, указанной в аргументе, всех строчных букв в прописные.

Формат вызова:

```
upper($f1)
```

Пример

```
# Необработанное событие:
# Example: Online_service
TEXT = 'Example: {$f1=STRING}'
datafield1 = upper($f1)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "ONLINE_SERVICE",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 48. Приведение типов данных функции upper

Аргумент	Результат
String	String

4.7. Функции для работы со списками

В разделе описаны функции языка XP для работы со списками и преобразования данных с типом List.

В этом разделе

4.7.1. append

Функция append используется для добавления в конец списка, указанного в первом аргументе, элемента, указанного во втором аргументе. Если первый аргумент функции не определен, функция возвращает список из одного элемента.

Формат вызова:

```
append($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example: rows
TEXT = 'Example: {$f2=REST?}'
$f1 = ["Delete the last ", 10]
$f3 = append($f1, $f2)
```

```
datafield1 = $f3[0] + string($f3[1]) + " " + $f3[2]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Delete the last 10 rows",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 49. Приведение типов данных функции append

Первый аргумент	Второй аргумент	Результат
List, Null	Любой	List

4.7.2. in_list

Функция `in_list` используется для поиска в списке. В первом аргументе функции указывается список с типом данных List, во втором аргументе — искомое значение. Если значение найдено в списке, функция возвращает True, в ином случае — False. При поиске учитывается тип данных аргументов.

Формат вызова:

```
in_list($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f1 = ["Red", "Green", "Orange"]
datafield1 = in_list($f1, "Green")
datafield2 = in_list($f1, "Blue")
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield1": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 50. Приведение типов данных функции in_list

Первый аргумент	Второй аргумент	Результат
List	Любой	Bool

4.7.3. is_list

Внимание! Функция может использоваться только при создании правил нормализации.

Функция `is_list` используется для проверки соответствия аргумента типу данных List. Если аргумент относится к типу данных List, функция возвращает значение True, в ином случае — False.

Формат вызова:

```
is_list($f1)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f = ["Red", "Green", "Orange"]
datafield1 = is_list($f)
datafield2 = is_list("Blue")
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "datafield2": "false",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 51. Приведение типов данных функции is_list

Аргумент	Результат
Любой	Bool

4.7.4. remove

Функция `remove` используется для удаления из списка, указанного в первом аргументе, всех элементов, значение которых совпадает со вторым аргументом.

Формат вызова:

```
remove($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f1 = ["Red", "Green", "Orange"]
$f3 = remove($f1, "Green")
datafield1 = in_list($f3, "Green")
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "false"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 52. Приведение типов данных функции remove

Первый аргумент	Второй аргумент	Результат
List	Любой, кроме Null, List	List

4.8. Функции для анализа данных

В разделе описаны функции языка XP для анализа данных.

В этом разделе

4.8.1. coalesce

Функция `coalesce` проверяет значения аргументов слева направо и возвращает значение первого отличного от null. Количество аргументов может быть любым. Если аргументы не заданы, функция возвращает null.

Формат вызова:

```
coalesce($f1, $f2, $f3,...)
```

Пример

```
# Необработанное событие:
# Example
TEXT = 'Example'
$f2 = "second"
$f3 = [3, "third"]
$f4 = 4
datafield1 = coalesce($f1, $f2, $f3, $f4)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"second",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 53. Приведение типов данных функции coalesce

Аргумент	Результат
Null	Null
Любой	Как у аргумента

4.8.2. flip_endianness16, flip_endianness32, flip_endianness64

Внимание! Функции могут использоваться только при создании правил нормализации.

Функции `flip_endianness16`, `flip_endianness32`, `flip_endianness64` используются для изменения порядка байтов 16-, 32- и 64-разрядных чисел на обратный.

Формат вызова:

```
flip_endianness16($f1)
flip_endianness32($f1)
flip_endianness64($f1)
```

Пример

```
# Необработанное событие:
# Example: 256
TEXT = 'Example: {$f1=NUMBER?}'
datafield1 = flip_endianness16($f1)
datafield2 = flip_endianness32($f1)
datafield3 = flip_endianness64($f1)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"1",
#    "datafield2":"65536",
#    "datafield3":"281474976710656",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 54. Приведение типов данных функций flip_endianness16, flip_endianness32, flip_endianness64

Аргумент	Результат
Number	Number

4.8.3. in_subnet

Функция `in_subnet` используется для проверки принадлежности IP-адреса к подсети. В первом аргументе указывается IP-адрес с типом данных IPAddress. Во втором аргументе указывается адрес подсети в формате CIDR-нотации в виде: `a.b.c.d/<Маска>` с типом данных String. Если один из аргументов не задан (null), функция возвращает значение False.

Формат вызова:

```
in_subnet($f1, $f2)
```

Пример

```
# Необработанное событие:
# Example: 10.0.2.3 10.0.0.0/22
TEXT = 'Example: {$f1=IPV4} {$f2=STRING}'
datafield1 = in_subnet($f1, $f2)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "true",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 55. Приведение типов данных функции in_subnet

Первый аргумент	Второй аргумент	Результат
IPAddress, Null	String	Bool

4.9. Функции для работы с табличными списками

В разделе описаны инструменты языка XP, которые вы можете использовать в правилах корреляции и обогащения для получения информации из табличных списков, созданных в MaxPatrol SIEM.

Для получения информации можно, либо выполнить инлайн-запрос, добавив к названию табличного списка префикс `table::`, либо с помощью функций `exec_query`, `select_query_first` обратиться к запросу, объявленному в правиле корреляции или обогащения в директиве `query`.

В этом разделе

4.9.1. Префикс table::, инлайн-запрос

Инлайн-запрос — запрос к табличному списку, условие которого указывается непосредственно в строке вызова запроса. Условие запроса проверяется для записей табличного списка, и если хотя бы для одной из записей условие выполнено, запрос возвращает True, в ином случае — False.

Для создания инлайн-запроса к табличному списку используется название этого табличного списка с префиксом `table::`. После названия в фигурных скобках нужно указать условие запроса.

Условие запроса может состоять из нескольких логических выражений, объединенных операторами `or`, `and`. В условии можно использовать круглые скобки и оператор `not`. В каждом выражении слева от логического оператора должны располагаться имена колонок табличного списка (с префиксом `column::`), справа от оператора можно использовать:

— поля события;

— литералы (с типом данных Bool, Null, Number, String);

— переменные (должны начинаться со знака доллара $);

— выражения языка XP с математическими операторами;

— функции языка XP.

Примечание. При использовании в условии запроса функции `in_subnet` в первом аргументе функции нужно указать имя переменной или поле события, во втором — имя колонки табличного списка (с префиксом `column::`).

Формат запроса:

```
table::<Название табличного списка> {
    <Условие запроса>
}
```

Пример

```
# Табличный список:
# "Tabular_List_Name": [
#    {"IP":"192.0.2.10", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.12", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
    key:
```

```
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    on Event{
        # Возвращает True, если порт в состоянии Open
        $datafield1 = table::Tabular_List_Name {
            column::IP == dst.ip and
            column::Port == dst.port and
            column::Status == "Open"
    }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"true",
#    ...}
```

4.9.2. Функция exec_query, запрос к табличному списку

> **Внимание!** Функция может использоваться только при создании правил корреляции и обогащения.

Функция `exec_query` используется для выполнения запроса к табличному списку. В первом аргументе нужно указать название запроса, описанного в директиве `query`. Во втором аргументе — список значений полей, по которым выполняется запрос. Количество передаваемых значений должно совпадать с количеством аргументов запроса.

Если для переданных значений полей условие в указанном запросе выполнено, функция возвращает True, в ином случае — False.

Формат вызова функции:

```
exec_query(<Название запроса>, [$f1, $f2, ...])
```

Формат запроса:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
}
```

Пример

```
# Табличный список:
# "Tabular_List_Name": [
#    {"IP":"192.0.2.10", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.12", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Запрос TestQuery к табличному списку Tabular_List_Name
query TestQuery ($check_ip, $check_port) from Tabular_List_Name {
     column::IP == $check_ip and
     column::Port == $check_port and
     column::Status == "Open"
}
event Event:
     key:
          event_src.title
     filter {
          event_src.title == "test"
     }
rule Example: Event
     on Event{
          # Возвращает True, если переданный в запрос порт в состоянии Open
          $datafield1 = exec_query("TestQuery", [dst.ip, dst.port])
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
```

```
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"true",
#    ...}
```

Таблица 56. Приведение типов данных функции exec_query

Первый аргумент	Второй аргумент	Результат
String	List = [String, String, ...]	Bool

4.9.3. Функция exec_query, арифметические операции над результатами запроса

> **Внимание!** Функция может использоваться только при создании правил корреляции и обогащения.

Функцию `exec_query` можно использовать для выполнения арифметических операций над элементами табличного списка, удовлетворяющих запросу.

В первом аргументе функции нужно указать название запроса, описанного в директиве `query`. В запросе нужно добавить инструкцию `qhandler`, в которой указать агрегатные функции (см. раздел 9.5.2) (например, `min`, `max`, `avg`) для выполнения арифметических операций над результатами запроса. Значение каждой функции нужно присвоить переменной, имя которой является ключом для получения значения. В аргументе каждой функции нужно указать имя колонки табличного списка (с префиксом `column::`), над элементами которого выполняется арифметическая операция. Поддерживаются операции над элементами колонок с типом данных Number.

Во втором аргументе функции нужно указать список значений полей, по которым выполняется запрос. Количество передаваемых значений должно совпадать с количеством аргументов запроса.

Если для переданных значений полей условие в указанном запросе выполнено, функция возвращает ассоциативный массив с типом данных KeyValue. Массив содержит результаты с типом данных Number для агрегатных функций, указанных в инструкции `qhandler`, над элементами указанной колонки из строк табличного списка, удовлетворяющих запросу. Для обращения к элементам массива в качестве ключей нужно использовать имена переменных инструкции `qhandler`.

Формат вызова функции:

```
$result_list = exec_query(<Название запроса к табличному списку>, [<Аргументы запроса>])
```

Формат обращения к элементам списка:

```
<Значение> = $result_list["<Ключ>"]
```

Формат запроса:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
}
     qhandler {
          <Ключ 1> = <Агрегатная функция 1>(column::<Имя колонки 1>)
          <Ключ 2> = <Агрегатная функция 2>(column::<Имя колонки 2>)
          ...
     }
```

Пример

```
# Табличный список "Tabular_List_Duration": [
#    {"IP":"192.0.2.10", "Port":22, "Status":"Open", "Duration":41},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":33},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":67}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Запрос TestQuery к табличному списку Tabular_List_Duration
query TestQuery ($check_ip) from Tabular_List_Duration {
     column::IP == $check_ip and
     column::Port == 22 and
     column::Status == "Open"
}
     qhandler {
          $get_max_duration = max(column::Duration)
          $get_min_duration = min(column::Duration)
          $get_avg_duration = avg(column::Duration)
```

```
     }
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
     on Event{
          # Присваивание переменным максимального, минимального и среднего времени
открытия порта 22 для переданного в запрос IP-адреса узла
          $get_duration = exec_query("TestQuery", [dst.ip])
          $datafield1 = "IP_" + string(dst.ip)
          $datafield2 = "Max_duration_" + string($get_duration["$get_max_duration"])
          $datafield3 = "Min_duration_" + string($get_duration["$get_min_duration"])
          $datafield4 = "Avg_duration_" + string($get_duration["$get_avg_duration"])
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"IP_192.0.2.11",
#    "datafield2":"Max_duration_67",
#    "datafield3":"Min_duration_33",
#    "datafield4":"Avg_duration_50",
#    ...}
```

Таблица 57. Приведение типов данных функции exec_query с инструкцией qhandler

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	List = [String, String, ...]	String	KeyValue

4.9.4. Функция select_query_first, запрос значения из табличного списка

> **Внимание!** Функция может использоваться только при создании правил корреляции и обогащения.

Функция `select_query_first` используется для получения значения ячейки табличного списка, удовлетворяющего запросу. В первом аргументе нужно указать название запроса, описанного в директиве `query`. Во втором аргументе — список значений полей, по которым выполняется запрос. Количество и тип данных передаваемых значений должно совпадать с количеством и типом данных аргументов запроса. В третьем аргументе нужно указать заголовок колонки табличного списка (в кавычках), значение которой требуется получить.

Если для переданных значений полей условие в указанном запросе выполнено, функция возвращает значение ячейки указанной колонки из первой строки табличного списка, удовлетворяющей запросу. Тип данных возвращаемого значения совпадает с типом данных колонки. Если условие запроса не выполнено, функция возвращает null.

Формат вызова функции:

```
select_query_first(<Название запроса>, [$f1, $f2, ...], "<Имя колонки>")
```

Формат запроса:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
}
```

Пример

```
# Табличный список:
# "Tabular_List_Name": [
#    {"IP":"192.0.2.10", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.12", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
```

```
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Запрос TestQuery к табличному списку Tabular_List_Name
query TestQuery ($check_ip, $check_port) from Tabular_List_Name {
     column::IP == $check_ip and
     column::Port == $check_port}
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
     on Event{
          # Присваивание переменной IP-адреса узла, если переданный в запрос порт на
этом узле в состоянии Open
          $datafield1 = "Port_Status: " + select_query_first("TestQuery", [dst.ip,
dst.port], "Status")
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"Port_Status: Open",
#    ...}
```

Таблица 58. Приведение типов данных функции exec_query_first

Первый аргумент	Второй аргумент	Второй аргумент	Результат
String	List = [String, String, ...]	String	Number, String, DateTime, Null (зависит от типа данных колонки табличного списка)

4.9.5. Функция regex_match, проверка строки по регулярному выражению из табличного списка

> **Внимание!** Функция может использоваться только при создании правил корреляции и обогащения.

Функция `regex_match` используется в запросе к табличному списку для сравнения строки с указанным в табличном списке шаблоном. Функцию можно использовать в инлайн-запросе или при описании запроса в директиве `query`. В первом аргументе функции указывается строка для сравнения с шаблоном, во втором — название колонки с шаблонами. Функция возвращает True, если строка соответствует одному из шаблонов в указанной колонке, в противном случае — False. Если первый аргумент не задан, функция возвращает False.

> **Примечание.** Вы можете выполнять запрос с функцией `regex_match` только к табличному списку типа «справочник». Для колонки с регулярными выражениями в табличном списке должен быть выбран тип Regexp. Поддерживаются Perl-совместимые регулярные выражения (PCRE).

Формат запроса:

```
query <Название запроса>($f) from <Название табличного списка> {
    regex_match($f, column::<Название колонки>)
}
```

или

```
table::<Название табличного списка> {
    regex_match($f, column::<Название колонки>)
}
```

Пример

```
# Табличный список:
# "Tabular_List_Service_Names": [
#    {"Object":".*first_service.exe"},
#    {"Object":".*second_service.exe"},
#    {"Object":".*third_service.exe"}]
# Нормализованное событие: {
#    "object.value":"path\folder\third_service.exe"
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
```

```
    #    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
    query TestQuery ($check_object) from Tabular_List_Service_Names {
        regex_match($check_object, column::Object)
    }
    event Event:
        key:
            event_src.title
        filter {
            event_src.title == "test" and
            exec_query("TestQuery", [object.value])
        }
    rule Example: Event
    on Event{
        $datafield1 = "Обнаружен объект " + object.value
    }
    emit {
        $correlation_type = "event"
        $object = "client"
        $action = "check"
        $status = "success"
        $importance = "info"
}
    # Корреляционное событие: {
    #    "correlation_type":"event",
    #    "object":"client",
    #    "action":"check",
    #    "status":"success",
    #    "importance":"info",
    #    "datafield1":"Обнаружен объект path\folder\third_service.exe",
    #    ...}
```

Таблица 59. Приведение типов данных функции regex_match

Первый аргумент	Второй аргумент	Результат
String	—	Bool
Null	—	Bool (False)

5. Создание правила нормализации

В этом разделе описаны ключевые слова, токены и функции, используемые при создании правил нормализации, даны примеры и рекомендации.

Для отладки нового правила вы можете использовать утилиты SDK или утилиту PTSIEMSDK GUI. Добавить новое правило в MaxPatrol SIEM вы можете через веб-интерфейс Knowledge Base (см. Руководство оператора).

В этом разделе

5.1. Алгоритм работы службы нормализации событий

При сборе событий MP 10 Collector, в зависимости от используемого модуля и метода сбора, для каждого необработанного события указывает его формат (в поле `mime`).

Алгоритм работы службы нормализации событий зависит от формата необработанного события. Для разных форматов используются специализированные алгоритмы обработки текста события (парсеры). Правила нормализации создаются с учетом особенностей работы алгоритмов обработки, форматы необработанных событий указываются с помощью ключевых слов форматной строки:

- TEXT — для неструктурированного текста, например стандарта syslog;
- TABULAR — для ассоциативного массива «ключ — значение» формата JSON;
- JSON — для формата JSON;
- EVENTLOG — для событий журнала Windows;
- XML — для событий формата XML.

Алгоритм обработки неструктурированного текста

В ходе обработки событий:

— Для каждого события по графу нормализации выполняется поиск подходящего правила нормализации. Текст события должен соответствовать шаблону, указанному в форматной строке (с ключевым словом `TEXT`). Если такое правило не найдено, событие не нормализуется.

— Выполняется последовательное распознавание текста необработанного события по шаблону, указанному в форматной строке.

— Если в правиле нормализации объявлено условие применения (в строке с ключевым словом `COND`), проверяется это условие. Если условие не выполнено, событие не нормализуется.

— Создается нормализованное событие, поля события заполняются значениями, указанными в блоке параметров правила нормализации.

Алгоритм обработки структурированного текста

В ходе обработки событий:

— Для каждого события по графу нормализации выполняется поиск правила нормализации с подходящей форматной строкой. Если такое правило не найдено, событие не нормализуется.

— Необработанные события журнала Windows и формата XML преобразуются в формат JSON.

— Параметры события заносятся в словарь в памяти службы нормализации событий.

— Если в правиле нормализации объявлено условие применения правила (в строке с ключевым словом `COND`), проверяется это условие. Если условие не выполнено, событие не нормализуется.

— Создается нормализованное событие, поля события заполняются значениями, указанными в блоке параметров правила нормализации.

См. также

Этапы обработки событий (см. раздел 2)

Программные компоненты для обработки событий (см. раздел 3)

5.2. Структура правила нормализации

Правило нормализации должно начинаться с ключевого слова, указывающего на формат необработанного события. После ключевого слова в одинарных или двойных кавычках указывается форматная строка, вид которой зависит от формата события.

Правило нормализации может содержать строку, начинающуюся с ключевого слова `COND` для объявления условия применения правила.

> **Примечание.** Ключевое слово форматной строки и слово `COND` могут начинаться с восклицательного знака (!). Такое написание считается устаревшим.

Правило нормализации может содержать функцию `submessage` — для объявления вложенного правила нормализации для распознавания фрагмента необработанного события и ключевые слова `subformula`, `endsubformula` — для ввода блока операторов вложенного правила нормализации.

В блоке операторов правила нормализации вы можете использовать функции, операторы и конструкции языка XP, переменные (должны начинаться с `$`) и поля нормализованного события.

```
<Ключевое слово> = '<Форматная строка>'
COND = (<Условие>)
submessage("<Формат фрагмента>", "<Название вложенного правила>", <Фрагмент события>)
subformula <Название вложенного правила>
    <Ключевое слово вложенного правила> = '<Форматная строка вложенного правила>'
    COND = (<Условие вложенного правила>)
    <Блок операторов вложенного правила>
endsubformula
<Блок операторов>
```

5.3. Заполнение полей нормализованного события

В правиле нормализации необходимо указать значения обязательных для заполнения полей нормализованного события (см. приложение Б). Кроме того, вы можете указать значения необязательных полей или изменить значения отдельных сервисных полей, заполняемых автоматически.

Заполнение полей нормализованного события выполняется в блоке операторов правила нормализации (и вложенного правила).

> **Примечание.** Для текстовых событий вы можете заполнять поля нормализованного события в форматной строке, присваивая им значения токенов внутри шаблонов.

В правиле нормализации нужно указать значения для следующих полей нормализованного события:

- `time` — для указания времени регистрации события на источнике;
- `event_src.title` — для указания названия продукта источника события; указывается в нижнем регистре без пробелов (можно использовать символ подчеркивания), например `windows`, `asa`, `oracle_database`;
- `event_src.vendor` — для указания производителя источника; указывается в нижнем регистре без пробелов, например `microsoft`, `cisco`;

— `action`, `object`, `status` — для описания характера, объекта и результата воздействия; могут принимать только определенные значения (см. приложение В);

Примечание. Вы можете заполнить необязательное поле `subject`, чтобы указать источник воздействия.

— `event_src.category` — для назначения категории источника события; может принимать только определенные значения (см. приложение Г);

— `importance` — для указания важности события для информационной безопасности; может принимать значения `info`, `low`, `medium`, `high`;

— `id` — для ввода идентификатора правила нормализации в виде: `<Пространство_имен>_<Производитель>_<Название_источника>_<Транспорт>_<Тип_события>`, например `PT_Cisco_IOS_syslog_SEC_LOGIN_QUIET_MODE_OFF`.

5.4. Ключевое слово COND. Условие использования правила нормализации

Ключевое слово `COND` используется для объявления условия применения правила нормализации (и вложенного правила). Если условие возвращает True, создается нормализованное событие, поля события заполняются данными согласно правилу нормализации.

Условие может содержать переменные, математические и логические операторы языка XP, значение null.

Формат вызова:

```
COND = <Условие>
```

или

```
COND = (
    <Условие>
    )
```

Пример

```
# Необработанное событие:
# Example: Air temperature 28 degrees
TEXT = 'Example: {WORD+}{$temperature=NUMBER}{WORD}'
COND = (number($temperature) >= 23)
datafield1 = "Too_hot_" + string($temperature) + "_degrees"
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Too_hot_28_degrees",
#    "time": "2017-10-18T12:00:00Z"}
```

5.5. Ключевое слово TEXT

Ключевое слово `TEXT` используется для объявления форматной строки события в формате простого текста. При нормализации текстового события выполняется последовательное распознавание слов и символов события для заполнения полей нормализованного события. Форматная строка в этом случае является шаблоном для распознавания текста события, она должна повторять его структуру и содержать шаблоны для распознавания отдельных слов или символов события (в фигурных скобках) и сами слова (символы), если их расположение в событии фиксировано.

> **Примечание.** Если форматная строка заключена в двойные кавычки, внутри шаблона слова нужно использовать одинарные кавычки.

Шаблон слова события может содержать токены языка XP, операторы токенов, любые символы (в двойных кавычках), переменные и поля нормализованного события. Элементы в шаблоне можно разделять пробелом.

Токен языка XP — лексема, определяющая специализированный алгоритм, который используется для распознавания последовательности символов в шаблоне слова.

Формат вызова:

```
TEXT = '{<Токен 1>}{<Токен 2>}...'
```

Пример

```
# Необработанное событие:
# Example: Temperature - 28 degrees; Pressure - 1 bar.
TEXT = 'Example: {WORD "-" $tmp = NUMBER}{WORD};
{WORD|WORDDASH}{"-"}{datafield2 = NUMBER}{WORD}{LITERAL}'
datafield1 = $tmp
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"28",
#    "datafield2":"1",
#    "time": "2017-10-18T12:00:00Z"}
```

В этом разделе

IPV6 (см. раздел 5.5.8)

KEYVALUE (см. раздел 5.5.9)

LITERAL (см. раздел 5.5.10)

MACADDR (см. раздел 5.5.11)

NTUSER (см. раздел 5.5.12)

NUMBER (см. раздел 5.5.13)

REST (см. раздел 5.5.14)

STRING (см. раздел 5.5.15)

UNTIL (см. раздел 5.5.16)

WORD (см. раздел 5.5.17)

WORDDASH (см. раздел 5.5.18)

5.5.1. Операторы токена

В шаблоне слова форматной строки текстового события можно использовать операторы токенов.

Оператор «=»

Результатом применения оператора «=» является присваивание переменной или полю нормализованного события значения токена.

Формат вызова:

```
TEXT='{$f=<Токен>}'
```

или

```
TEXT='{<Поле события>=<Токен>}'
```

Пример:

```
# Необработанное событие:
# Example: Hello
TEXT = 'Example: {datafield1 = WORD}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Hello",
#    "time": "2017-10-18T12:00:00Z"}
```

Оператор «+»

Результатом применения оператора «+» является использование токена для распознавания одной и более последовательностей символов, разделенных пробелами.

Формат вызова:

```
TEXT='{<Токен>+}'
```

Пример:

```
# Необработанное событие:
# Example: Hello wonderful world
TEXT = 'Example: {datafield1=WORD+}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Hello wonderful world",
#    "time": "2017-10-18T12:00:00Z"}
```

Оператор «*»

Результатом применения оператора «*» является использование токена для распознавания нескольких последовательностей символов, разделенных пробелами. В отличие от оператора «+» последовательность символов может отсутствовать в событии, это не приведет к ошибке распознавания.

Формат вызова:

```
TEXT='{<Токен>*}'
```

Пример:

```
# Необработанное событие:
# Example
TEXT = 'Example {datafield1=WORD*}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "time": "2017-10-18T12:00:00Z"}
```

Оператор «?»

Применение оператора «?» к токену делает необязательным его использование при распознавании события. Последовательность символов может отсутствовать в событии, это не приведет к ошибке распознавания.

Формат вызова:

```
TEXT='{<Токен>?}'
```

Пример:

```
# Необработанное событие:
# Example: Hello world !
TEXT = 'Example: {datafield1=WORD+}{datafield2=LITERAL?}'
time = "2017-10-18T12:00:00Z"
```

```
# Нормализованное событие: {
#    "datafield1": "Hello world",
#    "datafield2": "!",
#    "time": "2017-10-18T12:00:00Z"}
```

Операторы конъюнкции

Оператором конъюнкции (логическим И) в шаблоне слова форматной строки является пробел. Применение оператора позволяет использовать в одном шаблоне несколько токенов.

Формат вызова:

```
TEXT='{<Токен 1> <Токен2>}'
```

Пример:

```
# Необработанное событие:
# Example: Hello world !!!
TEXT = 'Example: {datafield1=WORD datafield2=WORD "!!!"}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"Hello",
#    "datafield2":"world",
#    "time": "2017-10-18T12:00:00Z"}
```

Оператор дизъюнкции «|»

Применение оператора дизъюнкции (логического ИЛИ) позволяет использовать один или другой токен, в зависимости от формата данных последовательности символов.

Формат вызова:

```
TEXT='{<Токен 1>|<Токен2>}'
```

При распознавании может использоваться токен 1 или токен 2.

```
TEXT='{<Токен 1>|<Токен2>|}'
```

При распознавании может использоваться токен 1, токен 2 или последовательность символов может отсутствовать в событии.

Пример:

```
# Необработанное событие:
# Example: 10.12.13.14
TEXT = 'Example: {datafield1=IPV4|datafield1=IPV6}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "10.12.13.14",
#    "time": "2017-10-18T12:00:00Z"}
```

5.5.2. BASE64

Токен BASE64 используется для распознавания строки, закодированной по стандарту Base64.

Токен имеет один аргумент, в котором можно указать тип определяемой последовательности Base64:

— STRICT — последовательности Base64, сформированные по стандарту RFC 4648.

— URL — последовательности типа Base64 URL.

— ALL — определение последовательностей Base64 обоих типов (STRICT и URL); используется по умолчанию.

Данные, полученные с помощью токена BASE64, имеют тип Buffer и при необходимости могут быть преобразованы в строку (тип String) с помощью функции decode.

Формат вызова:

```
TEXT = '{$f=BASE64("STRICT")}'
```

Пример

```
# Необработанное событие:
# Example: Q29tbW9uIEJBU0U2NA== U3RyaWN0IEJBU0U2NA== dXJsLXNhZmUgbWVzc2FnZQ
TEXT = 'Example: {$f1=BASE64} {$f2=BASE64("STRICT")} {$f3=BASE64("URL")}'
time = "2018-04-27T10:52:47Z"
datafield1 = decode(buffer_from_base64($f1), "UTF-8")
datafield2 = decode(buffer_from_base64($f2), "UTF-8")
datafield3 = decode(buffer_from_base64($f3), "UTF-8")
# Нормализованное событие: {
#    "time": "2018-04-27T10:52:47Z",
#    "datafield1": "Common BASE64",
#    "datafield2": "Strict BASE64",
#    "datafield3": "url-safe message"}
```

Таблица 60. Приведение типов данных токена BASE64

Аргумент	Результат
String	Buffer

5.5.3. CSV

Токен CSV используется для распознавания строки с CSV-разметкой и возвращает список элементов строки. Распознаются все символы до конца строки.

Токен имеет два аргумента: в первом нужно указать символ, разделяющий элементы строки, во втором — символ экранирования, стоящий перед каждым элементом строки и после него. Если элементы строки не экранированы, во втором аргументе нужно указать любой символ, не встречающийся в строке. Если токен используется без аргументов, считается, что элементы строки разделены запятой и не экранированы.

Формат вызова:

```
TEXT = '{$f=CSV}'
```

или

```
TEXT = '{$f=CSV("<Разделитель элементов>","<Экранирование элементов>")}'
```

Пример

```
# Необработанное событие:
# Example: /login=admin/; /password=supercleveradmin/
TEXT = 'Example: {$f=CSV(";","/")}'
datafield1 = $f[0]
datafield2 = $f[1]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "login=admin",
#    "datafield2": "password=supercleveradmin",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 61. Приведение типов данных токена CSV

Аргумент	Результат
String	List

5.5.4. DATETIME

Токен `DATETIME` используется для распознавания даты и времени и возвращает тип данных DateTime (формат ISO 8601). Формат даты и времени в событии определяется автоматически. Для сокращения времени распознавания вы можете указать имя формата из таблицы ниже.

Формат вызова:

```
TEXT = '{$f=DATETIME}'
```

или

```
TEXT = '{$f=DATETIME_<Имя формата>}'
```

Пример

```
# Необработанное событие:
# Example: 1990-10-31T23:13:04+03:00
TEXT = 'Example: {$f=DATETIME_ISO8601}'
```

```
datafield1 = $f
datafield2 = year($f)
datafield3 = month($f)
datafield4 = day($f)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "1990-10-31T23:13:04+03:00",
#    "datafield2": "1990",
#    "datafield3": "10",
#    "datafield4": "31",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 62. Приведение типов данных токена DATETIME

Аргумент	Результат
String	DateTime

Таблица 63. Форматы даты и времени токена DATETIME

Имя формата	Описание	Пример
CHECKPOINT_OPSEC	Встречается в ПО Check Point (часовой пояс не указан)	25Jan2015 01:01:01 25Jan2015 17:10:10
ISO8601	Соответствует ISO 8601	2015-01-25T12:12:12 2015-01-25T12:12:12Z 2015-01-25T12:12:12+03:00 2015-01-25T16:53:47.2146003+03:00
MMDDYYYY_HHMMSS	Разделителем между датой и временем может быть один любой символ. Разделителем внутри даты и времени может быть любой знак препинания. Разделителем между секундами и долями секунды может быть только точка (доли секунды — от 2 до 6 знаков)	01/25/2015 16:12:12 01-25-2015-16!12!12 01/25/2015 16:12:12.121 01/25/2015 16:12:12 MSK 01/25/2015 16:12:12 +03:00
SAP	Формат YYYYMMDDHHMMSS	20150125074639
SAP_PO	Встречается в ПО SAP Process Orchestration	2015 10 25 04:09:22:502#+0300

Имя формата	Описание	Пример
SEDO	Встречается в системе электронного документооборота «Электронная Москва»	3/Jul/2015:11:02:47 +0300 03/Jul/2015:11:02:47 +0300 25/Jul/2015:11:02:47 +0300 25/Jul/2015:08:02:47
SYSLOG	Встречается в сетевых устройствах и ОС семейства Unix. Используются сокращения для часовых поясов: MSK — UTC+03:00; UTC, GMT и остальные — UTC+00:00	Jan 25 01:00:00 Jan 25 16:12:12 Jan 25 16:12:12 MSK Jan 25 2015 16:12:12 Jan 25 2015 16:12:12 MSK Jan 25 16:12:12.133 Jan 25 16:12:12.133 MSK Jan 25 16:12:12.133331 Jan 25 16:12:12.133331 MSK Jan 25 2015 16:12:12.133331 Jan 25 2015 16:12:12.133331 MSK
SYSLOG_HUAWEI	Встречается в сетевых устройствах Huawei и Juniper	Jun 25 02:29:19 2000 Jan 25 16:50:56:487 2011 Jan 25 16:50:56.487 2011 Jan 25 16:50:56,487 2011
SYSLOG_NEXUS	Встречается в сетевых устройствах Cisco Nexus	2015 Aug 2 13:10:44 MSK 2015 Aug 25 16:51:55 UTC 2015 Aug 25 17:45:23 2015 Aug 25 16:51:55.123 UTC 2015 Aug 25 17:45:23.123
WINDOWS_CIM	Формат времени параметра CIM_DATETIME из WMI вида YYYYMMDDHHMMSS.mmmmmmsUUU	20150125173254.006182+180

Имя формата	Описание	Пример
YYYYMMDD_HHMMSS	Встречается в ПО компании «Лаборатория Касперского». Разделителем между датой и временем может быть любой символ, кроме буквы «T»	2015/06/25 16:12:12 2015-06-25-16!12!12 2015/06/25@16:12:12.121 2015-06-25 12:10:14.4707

5.5.5. DURATION

Токен DURATION используется для преобразования строки, содержащей интервал времени в формате XXXd XXh:XXm:XXs или [HHH]HH:MM:SS, в число секунд.

Формат вызова:

```
TEXT = '{$f=DURATION}'
```

Пример

```
# Необработанное событие:
# Example: 0d 1h:2m:15s
TEXT = 'Example: {$f=DURATION}'
datafield1 = div($f,60*60)
datafield2 = div(mod($f,3600),60)
datafield3 = mod(mod($f,3600),60)
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"1",
#    "datafield2":"2",
#    "datafield3":"15",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 64. Приведение типов данных токена DURATION

Аргумент	Результат
String	Number

5.5.6. HOSTNAME

Токен HOSTNAME используется для распознавания имени узла в виде последовательности букв и цифр, которые могут быть разделены точками, знаками подчеркивания (_) и дефисами (-).

Формат вызова:

```
TEXT = '{$f=HOSTNAME}'
```

Пример

```
# Необработанное событие:
# Example: Logon Some-Host-Name_com.co
TEXT = 'Example: {WORD+}{datafield1=HOSTNAME}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"Some-Host-Name_com.co",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 65. Приведение типов данных токена HOSTNAME

Аргумент	Результат
String	String

5.5.7. IPV4

Токен `IPV4` используется для распознавания IP-адреса стандарта IPv4 и преобразования его в тип данных IPAddress.

Формат вызова:

```
TEXT = '{$f=IPV4}'
```

Пример

```
# Необработанное событие:
# Example: 192.168.0.1
TEXT = 'Example: {datafield1=IPV4}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"192.168.0.1"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 66. Приведение типов данных токена IPV4

Аргумент	Результат
String	IPAddress

5.5.8. IPV6

Токен `IPV6` используется для распознавания IP-адреса стандарта IPv6 и преобразования его в тип данных IPAddress.

Поддерживаются IP-адреса в форматах X:X:X:X:X:X:X.X, X:X:X:X:X:X:D.D.D.D и X:X:X:X:X:X:X.X%S, где X — шестнадцатеричные значения для 16-битовых частей адреса, D — десятичные значения для 8-битовых частей адреса (стандарта IPv4), S — четырехзначное десятичное число. Для замены одного или нескольких нулей в расположенных рядом 16-битовых частях адреса допускается использование двух двоеточий (::).

Формат вызова:

```
TEXT = '{$f=IPV6}'
```

Пример

```
# Необработанное событие:
# Example: 211:DB8:0:0:8:800:200C:47A 211:DB8:0:0:8:800:192.168.0.1
211:DB8:0:0:8:800:200C:47A%5141
TEXT = 'Example: {event_src.host=IPV6} {src.host=IPV6} {dst.host=IPV6}'
time = "2021-10-18T12:00:00Z"
# Нормализованное событие: {
#    "event_src.host": "211:DB8:0:0:8:800:200C:47A",
#    "src.host": "211:DB8:0:0:8:800:192.168.0.1",
#    "dst.host": "211:DB8:0:0:8:800:200C:47A",
#    "time": "2021-10-18T12:00:00Z"}
```

Таблица 67. Приведение типов данных токена IPV6

Аргумент	Результат
String	IPAddress

5.5.9. KEYVALUE

Токен KEYVALUE используется для распознавания ассоциативного массива, содержащего пары «ключ — значение». Распознаются все символы до конца строки.

Токен имеет три аргумента: в первом нужно указать символ, разделяющий элементы строки, во втором — символ, разделяющий пару «ключ — значение», в третьем — символ экранирования значения, стоящий перед каждым значением в паре и после него.

> **Примечание.** Для ввод одинарных кавычек используйте обратную косую черту (\), для ввода двойных кавычек — две обратные косые черты (\\).

Если токен используется без аргументов, считается, что элементы строки разделены запятой, пары «ключ — значение» разделены знаком равенства (=) и не экранированы.

Формат вызова:

```
TEXT = '{$f1=KEYVALUE}'
```

или

```
TEXT = '{$f1=KEYVALUE("<Разделитель между парами>","<Разделитель в паре>",
"<Экранирование значения>")}'
```

Пример

```
# Необработанное событие:
# Example: login:'admin'/password:'supercleveradmin'
TEXT = 'Example: {$f=KEYVALUE("/",":","\'")}'
datafield1 = $f["login"]
datafield2 = $f["password"]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"admin",
#    "datafield2": "supercleveradmin",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 68. Приведение типов данных токена KEYVALUE

Аргумент	Результат
String	KeyValue

5.5.10. LITERAL

Токен `LITERAL` используется для распознавания одного символа. Токен можно использовать для распознавания символов не разделенных пробелами.

Формат вызова:

```
TEXT = '{$f=LITERAL}'
```

Пример

```
# Необработанное событие:
# Example: Hello world !!!
TEXT = 'Example: {WORD WORD}{datafield1=LITERAL+}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "! ! !",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 69. Приведение типов данных токена LITERAL

Аргумент	Результат
String	String

5.5.11. MACADDR

Токен `MACADDR` используется для распознавания MAC-адреса и преобразования его в тип данных MACAddress.

Формат вызова:

```
TEXT = '{$f=MACADDR}'
```

Пример

```
# Необработанное событие:
# Example: 00:0C:29:27:6E:D0
TEXT = 'Example: {datafield1=MACADDR}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "00:0C:29:27:6E:D0",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 70. Приведение типов данных токена MACADDR

Аргумент	Результат
String	MACAddress

5.5.12. NTUSER

Токен `NTUSER` используется для распознавания имени пользователя в формате `<Имя домена>\<Имя пользователя>`. Имя может содержать цифры, буквы и обратную косую черту (\).

Формат вызова:

```
TEXT = '{$f=NTUSER}'
```

Пример

```
# Необработанное событие:
# Example: This is AC\\DC
TEXT = 'Example: {WORD+}{datafield1=NTUSER}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "AC\\\\DC"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 71. Приведение типов данных токена NTUSER

Аргумент	Результат
String	String

5.5.13. NUMBER

Токен `NUMBER` используется для распознавания целого числа.

Формат вызова:

```
TEXT = '{$f=NUMBER}'
```

Пример

```
# Необработанное событие:
# Example: Route 66
TEXT = 'Example: {WORD+}{datafield1=NUMBER}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "66",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 72. Приведение типов данных токена NUMBER

Аргумент	Результат
String	Number

5.5.14. REST

Токен `REST` используется для распознавания последовательности символов и пробелов до конца строки. Если символов не найдено, возвращает пустое значение.

Формат вызова:

```
TEXT = '{$f=REST}'
```

Пример

```
# Необработанное событие:
# Example: System check is finished!
TEXT = 'Example: {WORD WORD}{datafield1=REST}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "is finished!",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 73. Приведение типов данных токена REST

Аргумент	Результат
String	String

5.5.15. STRING

Токен `STRING` используется для распознавания последовательности символов до пробела.

Формат вызова:

```
TEXT = '{$f=STRING}'
```

Пример

```
# Необработанное событие:
# Password: As4r2@6E
TEXT = '{WORD}:{datafield1=STRING}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"As4r2@6E"}
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 74. Приведение типов данных токена STRING

Аргумент	Результат
String	String

5.5.16. UNTIL

Токен `UNTIL` используется для распознавания последовательности символов (и пробелов) до ограничителя, указанного в аргументе токена. Ограничителем может быть любая последовательность символов, ограничитель не попадает в возвращаемое токеном значение.

Примечание. Вы можете применить к токену только оператор «?».

Формат вызова:

```
TEXT = '{$f=UNTIL("<Ограничитель>")}'
```

Пример

```
# Необработанное событие:
# Example: System check is finished
TEXT = 'Example: {datafield1=UNTIL("check is")}{datafield2=REST}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"System ",
#    "datafield2":"finished"
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 75. Приведение типов данных токена UNTIL

Аргумент	Результат
String	String

5.5.17. WORD

Токен `WORD` используется для распознавания отдельных слов, разделенных пробелами. Слова могут состоять из последовательности символов, состоящей из букв, цифр и знака подчеркивания.

Формат вызова:

```
TEXT = '{$f=WORD}'
```

Пример

```
# Необработанное событие:
# Example: System_check is finished
TEXT = 'Example: {datafield1=WORD}{datafield2=WORD*}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"System_check"
#    "datafield2":"is finished",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 76. Приведение типов данных токена WORD

Аргумент	Результат
String	String

5.5.18. WORDDASH

Токен `WORDDASH` используется для распознавания отдельных слов, разделенных пробелами. Слова могут состоять из последовательности символов, включающей буквы, цифры, дефисы и знаки подчеркивания.

Формат вызова:

```
TEXT = '{$f=WORDDASH}'
```

Пример

```
# Необработанное событие:
# Example: S-y-s-t-e-m check_is finished
TEXT = 'Example: {datafield1=WORDDASH}{datafield2=WORDDASH}{datafield3=WORDDASH*}'
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1":"S-y-s-t-e-m",
#    "datafield2":"check_is",
#    "datafield3":"finished",
#    "time": "2017-10-18T12:00:00Z"}
```

Таблица 77. Приведение типов данных токена WORDDASH

Аргумент	Результат
String	String

5.6. Ключевое слово TABULAR

Ключевое слово TABULAR используется для объявления форматной строки правила нормализации событий, представленных в виде ассоциативного массив в формате JSON, например:

```
{
    "<Ключ 1>": "<Значение 1>",
    "<Ключ 2>": "<Значение 2>",
    "<Ключ 3>": [<Значение 31>, <Значение 32>, <Значение 33>]
}
```

При поступлении события в службу нормализации параметры события заносятся в словарь в памяти службы в виде ассоциативного массива, повторяющего структуру необработанного события. Значения параметров могут иметь типы данных Bool, String, Number или List.

Синтаксис форматной строки

В форматной строке вы можете указать параметры, которые должны находиться в событии (не равны null) для использования правила нормализации, или оставить ее пустой. Вы можете присвоить значения этих параметров переменным или полям нормализованного события.

> **Примечание.** С помощью ключевого слова COND вы можете объявить дополнительное условие использования правила нормализации.

Формат вызова:

```
TABULAR = ''
```

или

```
TABULAR = '<Ключ 1>,<Ключ 2>,...,{<Название поля>=<Ключ N-1>},{$f=<Ключ N>}'
```

Обращение к словарю параметров

Из блока операторов правила нормализации или из условия COND вы можете обратиться к словарю для получения значения параметра необработанного события по его ключу. При указании ключа его нужно начинать со знака доллара ($).

Формат обращения к словарю:

```
$f=$<Ключ>
```

или

```
$f=$<Ключ>[Номер элемента списка]
```

если ключ содержит спецсимволы:

```
$f=$['<Ключ>']
```

Примечание. Отсчет элементов списка начинается от нуля, то есть для обращения к первому элементу нужно указать 0, ко второму — 1 и так далее.

Пример

```
# Необработанное событие:
# {"firstName":"Vasily",
# "lastName":"Ivanov",
# "birthDate":"1976-08-23T00:00:00",
# "phoneNumbers":["812 123-1234","991 123-4567"],
# "user_id/user_code":"74328042/VI43"}
TABULAR = '{datafield1=firstName}, phoneNumbers'
COND = ($lastName=="Ivanov" and month(datetime($birthDate))>=5)
datafield2 = $phoneNumbers[0]
datafield3 = $phoneNumbers[1]
datafield4 = $['user_id/user_code']
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "Vasily",
#    "datafield2": "812 123-1234",
#    "datafield3": "991 123-4567",
#    "datafield4": "74328042/VI43",
#    "time": "2017-10-18T12:00:00Z"}
```

5.7. Ключевое слово JSON

Ключевое слово `JSON` используется для объявления форматной строки правила нормализации событий в формате JSON. Событие представляет собой вложенную структуру параметров (многоуровневый ассоциативный массив), например:

```
{
    "<Ключ 11>": "<Значение 11>",
    "<Ключ 12>": {
        "<Ключ 21>": "<Значение 21>",
        "<Ключ 22>": {
            <Ключ 31>: <Значение 31>,
            <Ключ 32>: [<Значение 321>, <Значение 322>, <Значение 323>]
            },
        "<Ключ 23>": "<Значение 23>"
        }
    "<Ключ 13>": "<Значение 13>"
}
```

При поступлении события в службу нормализации параметры события заносятся в словарь в памяти службы в виде многоуровневого ассоциативного массива, повторяющего структуру необработанного события. Значения параметров могут иметь типы данных Bool, String, Number или List.

Синтаксис форматной строки

В форматной строке вы можете указать одно условие использования правила нормализации или оставить ее пустой. С помощью условия в форматной строке вы можете проверить наличие параметра в событии или проверить его значение с типом данных Bool, String или Number (обращение к элементам списков не поддерживается). При указании составного ключа параметра используется точка.

> **Примечание.** С помощью ключевого слова COND вы можете объявить дополнительное условие использования правила нормализации.

Формат вызова:

```
JSON = ''
```

или

```
JSON = '<Ключ 1x>.<Ключ 2x>....<Ключ Nx>'
```

или

```
JSON = '<Ключ 1x>.<Ключ 2x>....<Ключ Nx>="<Значение Nx>"'
```

или

```
JSON = '<Ключ 1x>.<Ключ 3x>....<Ключ Nx>=<Значение Nx>'
```

Обращение к словарю параметров

Из блока операторов правила нормализации или из условия COND вы можете обратиться к словарю для получения значения параметра необработанного события по его ключу. При указании ключа его нужно начинать со знака доллара ($). При указании составного ключа используются квадратные скобки (как при обращении к ассоциативному массиву).

Формат обращения к словарю:

```
$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"]
```

или

```
$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"][<Номер элемента списка>]
```

если ключ содержит спецсимволы:

```
$f=$['<Ключ>']
```

> **Примечание.** Отсчет элементов списка начинается от нуля, то есть для обращения к первому элементу нужно указать 0, ко второму — 1 и так далее.

Пример

```
# Необработанное событие:
# {"firstName": "Vasily",
# "lastName": "Ivanov",
# "birthDate": {"year": 1976, "month": 8, "day": null},
# "phoneNumbers": ["812 123-1234", "991 123-4567"],
# "user_id/user_code": "74328042/VI43"}
JSON = 'birthDate.year=1976'
COND = ($lastName=="Ivanov" and $birthDate["month"]>=5)
datafield1 = $phoneNumbers[0]
datafield2 = $phoneNumbers[1]
datafield3 = $['user_id/user_code']
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "812 123-1234",
#    "datafield2": "991 123-4567",
#    "datafield3": "74328042/VI43",
#    "time": "2017-10-18T12:00:00Z"}
```

5.8. Ключевое слово EVENTLOG

Ключевое слово `EVENTLOG` используется для объявления форматной строки правила нормализации событий из журнала Windows.

При получении MP 10 Collector события автоматически преобразуются в формат JSON, например:

```
{
    "Event": {
        "xmlns": "<Адрес пространства имен XML>",
        "System": {
            "Provider": {
                "Name": "<Имя провайдера>"
            },
            "EventID": {
                "text": "<Идентификатор события>",
                "Qualifiers": "0"
                },
            "Level": "4",
            "Task": "0",
            "Keywords": "0x80000000000000",
            "TimeCreated": {
                "SystemTime": "<Дата и время регистрации>"
            },
            "EventRecordID": "72365",
            ...
```

```
            "Channel": "<Название журнала>",
            "Computer": "<Имя узла>",
            "Security": {
                "UserID": "S-1-5-18"
            }
        },
        "EventData": {
            "Data": [
                {"Name":"<Имя 1>",
                "text":"<Значение 1>"},
                {"Name":"Имя 2",
                "text":"Значение 2"},
                {"Name":"Имя 3",
                "text":"Значение 3"},
                ],
            ...
        }
    }
}
```

При поступлении события в службу нормализации событий выполняется автоматическая обработка события для упрощения его структуры. Из события удаляется общий для всех параметров ключ Event, удаляются ключи EventData и System, а относящиеся к ним параметры заносятся в словарь в виде многоуровневого ассоциативного массива. Если параметр Data содержит вложенные параметры Name — text, то они преобразуются в ассоциативный массив «ключ — значение». Значения параметров в словаре могут иметь типы данных Bool, String, Number или List. Структура параметров в словаре для указанного выше события примет вид:

```
{
    "Provider": {
        "Name": "<Имя провайдера>"
    },
    "EventID": {
        "text": "<Идентификатор события>",
        "Qualifiers": "0"
        },
    "Level": "4",
    "Task": "0",
    "Keywords": "0x80000000000000",
    "TimeCreated": {
        "SystemTime": "<Дата и время регистрации>"
    },
    "EventRecordID": "72365",
    ...
    "Channel": "<Название журнала>",
    "Computer": "<Имя узла>",
    "Security": {
        "UserID": "S-1-5-18"
```

```
    },
    "Data": [
        "<Имя 1>":"<Значение 1>",
        "<Имя 2>":"<Значение 2>",
        "<Имя 3>":"<Значение 3>"
     ]
}
```

Синтаксис форматной строки

> **Внимание!** При указании ключей в правиле нормализации (в том числе в форматной строке) нужно использовать упрощенную структуру параметров из словаря, а не из исходного JSON-события.

В форматной строке вы можете указать одно условие использования правила нормализации или оставить ее пустой. С помощью условия в форматной строке вы можете проверить наличие параметра в событии или проверить его значение с типом данных Bool, String или Number (обращение к элементам списков не поддерживается). При указании составного ключа параметра используется точка.

В качестве условия нормализации события рекомендуется использовать идентификатор события, указанный в параметре EventID.text (EventID).

> **Примечание.** С помощью ключевого слова COND вы можете объявить дополнительное условие использования правила нормализации.

Формат вызова:

```
EVENTLOG = 'EventID.text="<Идентификатор события>"'
```

или

```
EVENTLOG = ''
```

или

```
EVENTLOG = '<Ключ 1x>.<Ключ 2x>....<Ключ Nx>'
```

или

```
EVENTLOG = '<Ключ 1x>.<Ключ 2x>....<Ключ Nx>="<Значение Nx>"'
```

или

```
EVENTLOG = '<Ключ 1x>.<Ключ 3x>....<Ключ Nx>=<Значение Nx>'
```

Обращение к словарю параметров

Из блока операторов правила нормализации или из условия COND вы можете обратиться к словарю для получения значения параметра необработанного события по его ключу. При указании ключа его нужно начинать со знака доллара ($). При указании составного ключа используются квадратные скобки (как при обращении к ассоциативному массиву).

Формат обращения к словарю:

```
$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"]
```

или

```
$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"][Номер элемента списка]
```

Примечание. Отсчет элементов списка начинается от нуля, то есть для обращения к первому элементу нужно указать 0, ко второму — 1 и так далее.

Пример

```
# Необработанное событие:
# {"Event": {
#    "System": {
#      "Provider": {
#        "Name": "Developer Guide"
#      },
#      "EventID": {
#        "Qualifiers": "0",
#        "text": "777"
#      },
#      "TimeCreated": {
#        "SystemTime": "2016-04-14T16:35:40.000000000Z"
#      }
#    },
#    "EventData": {
#      "Data": [{
#          "Name": "firstName",
#          "text": "Vasily"
#        }, {
#          "Name": "lastName",
#          "text": "Ivanov"
#        }, {
#          "Name": "birthDate",
#          "text": "1976-08-23T00:00:00"
#        }, {
#          "Name": "phoneNumbers",
#          "text": ["812 123-1234", "991 123-4567"]
#        }
#      ]
#    }
#  }
# }
EVENTLOG = 'EventID.text="777"'
COND = ($Data["lastName"]=="Ivanov" and
    month(datetime($Data["birthDate"]))>=5)
datafield1 = $Data["phoneNumbers"][0]
```

```
datafield2 = $Data["phoneNumbers"][1]
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "812 123-1234",
#    "datafield2": "991 123-4567",
#    "time": "2017-10-18T12:00:00Z"}
```

5.9. Ключевое слово XML

Ключевое слово XML используется для объявления форматной строки правила нормализации событий в формате XML. Событие представляет собой вложенную структуру параметров (многоуровневый ассоциативный массив), например:

```
<<Ключ 1>>
    <<Ключ 21>><Значение 21><</Ключ 21>
    <<Ключ 22> <Имя атрибута 221>=<Значение атрибута 221> <Имя атрибута
222>=<Значение атрибута 222>><Значение 22><</Ключ 22>
    <<Ключ 23>>
        <<Ключ 31>><Значение 31><</Ключ 31>>
        <<Ключ 32>><Значение 321><</Ключ 32>>
        <<Ключ 32>><Значение 322><</Ключ 32>>
    <</Ключ 23>>
<</Ключ 1>>
```

При поступлении события в службу нормализации выполняется автоматическая обработка события. Для каждого параметра создается ассоциативный массив, в который заносятся значение параметра с ключом ! и атрибуты параметра, ключами которых являются их названия с добавленным в начале названия символом @. Параметры события заносятся в словарь в виде многоуровневого ассоциативного массива в формате JSON, повторяющего структуру необработанного события. Значения параметров могут иметь тип данных String.

```
{
    "<Ключ 1>":{
        "<Ключ 21>":{
            "!":"<Значение 21>"
        },
        "<Ключ 22>":{
            "!":<Значение 22>,
            "@<Имя атрибута 221>":"<Значение атрибута 221>",
            "@<Имя атрибута 222>":"<Значение атрибута 222>"
        },
        "<Ключ 23>":{
           "<Ключ 31>":{
                "!":"<Значение 31>"
            },
            "<Ключ 32>":{
                !":"<Значение 32>"
            },
```

```
            "<Ключ 32>":{
                "!":"<Значение 32>"
            },
        }
    }
}
```

Синтаксис форматной строки

В форматной строке вы можете указать параметр, который должен находиться в событии для использования правила нормализации, или оставить ее пустой. При указании составного ключа параметра используется точка.

> **Примечание.** С помощью ключевого слова COND вы можете объявить дополнительное условие использования правила нормализации.

Формат вызова:

```
XML = ''
```

или

```
XML = '<Ключ 1x>.<Ключ 2x>....<Ключ Nx>'
```

Обращение к словарю параметров

Из блока операторов правила нормализации или из условия COND вы можете обратиться к словарю для получения значения параметра необработанного события по его ключу. При указании ключа его нужно начинать со знака доллара ($). При указании составного ключа используются квадратные скобки (как при обращении к ассоциативному массиву).

Формат обращения к словарю:

`$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"]["!"]` — получение значения параметра,

`$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"][@<Имя атрибута Nx>]` — получение значения атрибута параметра,

`$f=$<Ключ 1x>["<Ключ 2x>"]...["<Ключ Nx>"][@<Номер параметра в списке>]` — получение значения параметра в списке параметров с одинаковыми названиями. При обращении к первому элементу в списке номер можно не указывать.

> **Примечание.** Отсчет элементов списка начинается от нуля, то есть для обращения к первому элементу нужно указать 0, ко второму — 1 и так далее.

Альтернативный формат обращения к словарю:

`$f=$<Ключ 1x>/<Ключ 2x>/.../<Ключ Nx>` — получение значения параметра,

`$f=$<Ключ 1x>/<Ключ 2x>/.../<Ключ Nx>/@<Имя атрибута Nx>` — получение значения атрибута параметра,

`$f=$<Ключ 1x>/<Ключ 2x>/.../<Ключ Nx>/<Номер параметра в списке>` — получение значения параметра в списке параметров с одинаковыми названиями. При обращении к первому элементу в списке номер можно не указывать.

Пример

```
# Необработанное событие:
# <user>
#    <firstName>Vasily</firstName>
#    <lastName>Ivanov</lastName>
#    <birthDate>
#        <year>1976</year>
#        <month>8</month>
#        <day>1</day>
#    </birthDate>
#    <phoneNumbers>
#        <number type="landline">812 123-1234</number>
#        <number type="cell">991 123-4567</number>
#    </phoneNumbers>
# </user>
XML = "user.birthDate.year"
COND = $user["lastName"]["!"] == "Ivanov"
    and number($user["birthDate"]["month"]["!"]) >= 5
datafield1 = $user["phoneNumbers"]["number"][0][@type]
datafield2 = $user["phoneNumbers"]["number"][0]
datafield3 = $user/phoneNumbers/number/1/@type
datafield4 = $user/phoneNumbers/number/1
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "landline",
#    "datafield2": "812 123-1234",
#    "datafield3": "cell",
#    "datafield4": "991 123-4567",
#    "time": "2017-10-18T12:00:00Z"}
```

5.10. Команда drop

Команда `drop` используется для прекращения нормализации события. Нормализованное событие не создается.

Формат вызова:

```
drop
```

Пример

```
# Необработанное событие:
# Example: Click 2 times
TEXT = 'Example: {STRING} {$f=NUMBER} {STRING}'
if $f == 1 then
    datafield1 = "Click again"
elif $f == 2 then
    datafield1 = "OK"
else
    drop
endif
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "datafield1": "OK",
#    "time": "2017-10-18T12:00:00Z"}
```

5.11. Вложенное правило нормализации

В правило нормализации событий вы можете добавить несколько вложенных правил для распознавания фрагментов необработанного события, имеющих другой формат. Для создания вложенного правила нормализации нужно объявить его с помощью функции `submessage` и ввести блок операторов правила внутри конструкции из ключевых слов `subformula`, `endsubformula`.

Пример

```
# Необработанное событие:
# {"timegenerated":"2016-05-22 08:44:26.640",
# "userID":334,
# "userHost":"10.19.15.24",
# "UserData":"<user><firstName>Vasily</firstName><lastName>Ivanov</
lastName><birthDate><year>1976</year><month>8</month><day>1</day></
birthDate><phoneNumbers><number type=\"landline\">812 123-1234</number><number
type=\"cell\">991 123-4567</number></phoneNumbers></user>"}
TABULAR = 'UserData, {time=timegenerated},{event_src.hostname=userHost}'
submessage("XML","Example_subformula",$UserData)
subformula 'Example_subformula'
    XML = 'user.birthDate.year'
    COND = $user["lastName"]["!"] == "Ivanov"
    datafield1 = $user["phoneNumbers"]["number"][0]
    datafield2 = $user/phoneNumbers/number/1
endsubformula
time = "2017-10-18T12:00:00Z"
# Нормализованное событие: {
#    "event_src.hostname": "10.19.15.24",
#    "datafield1": "812 123-1234",
```

```
#    "datafield2": "991 123-4567",
#    "time": "2017-10-18T12:00:00Z"}
```

В этом разделе

5.11.1. submessage

Функция `submessage` используется для объявления вложенного правила нормализации для распознавания фрагмента необработанного события, имеющего другой формат. В первом аргументе нужно указать ключевое слово для формата фрагмента события, во втором — название вложенного правила нормализации, в третьем — имя переменной с фрагментом исходного события для распознавания с помощью вложенного правила нормализации.

Формат вызова:

```
submessage("<Формат фрагмента>", "<Название вложенного правила>", $fragment)
```

Таблица 78. Приведение типов данных функции submessage

Первый аргумент	Второй аргумент	Третий аргумент	Результат
String	String	String	—

5.11.2. subformula ... endsubformula

Конструкция из ключевых слов `subformula`, `endsubformula` используется для ограничения вложенного правила нормализации. В блоке операторов вложенного правила вы можете использовать функции, операторы и конструкции языка XP, переменные (должны начинаться с $) и поля нормализованного события.

Формат вызова:

```
subformula '<Название вложенного правила>'
    <Ключевое слово вложенного правила> = '<Форматная строка вложенного правила>'
    COND = (<Условие вложенного правила>)
    <Блок операторов вложенного правила>
endsubformula
```

Таблица 79. Приведение типов данных функции subformula

Аргумент	Результат
String	—

6. Создание макроса

В этом разделе описана структура макроса. Макрос представляет собой фрагмент кода, сохраненный в отдельном файле `<Название макроса>.flt`.

Добавить новый макрос в MaxPatrol SIEM вы можете через веб-интерфейс Knowledge Base (см. Руководство оператора).

Вы можете вызывать макросы из условия отбора событий в инструкции `filter` директивы `event` в правилах обогащения и корреляции. Для вызова макроса нужно использовать его название с префиксом `filter::`.

Структура макроса

Макрос определяется директивой `filter`. Определение макроса состоит из ключевого слова `filter`, за которым следуют список параметров и условие макроса в фигурных скобках.

```
filter <Название макроса>(<Параметры макроса>){
    <Условие макроса>
}
```

Формат вызова макроса:

```
filter::<Название макроса>(<Параметры макроса>)
```

Директива filter

Директива `filter` предназначена для определения макроса. В директиве нужно указать название, параметры и условие макроса.

Название макроса может состоять из букв латинского алфавита, цифр и знака подчеркивания, должно начинаться с прописной буквы и быть уникальным в рамках всего набора названий макросов.

Параметры макроса используются при составлении условия макроса и при вызове макроса заменяются указанными аргументами (например, полями события, математическими или логическими выражениями). Имена параметров должны начинаться со знака доллара ($). Параметры перечисляются через запятую, перед каждым параметром должен быть указан тип данных (`bool`, `number` или `string`). Макрос может не содержать параметров.

Условие макроса может состоять из нескольких логических выражений, объединенных операторами `or`, `and`. Если при проверке условие возвращает значение True, вызов макроса также возвращает True, в ином случае — False. В логических выражениях можно использовать:

— математические и логические операторы;

— поля события;

— литералы (с типом данных Bool, Null, Number, String);

— параметры макроса (должны начинаться со знака доллара $);

— инлайн-запросы к табличным спискам (с префиксом `table::`);

— вызовы других макросов, рекурсивные вызовы не допускаются (с префиксом `filter::`);

— функции языка XP.

Пример

```
filter Example(string $action){
    action == $action
    and table::Example_TableName {
       (column::username == subject.name and column::domain == null)
       or (column::username == subject.name and column::domain == subject.domain)
    }
}
```

См. также

Префикс table::, инлайн-запрос (см. раздел 4.9.1)

7. Создание шаблона исключений для заполнения табличного списка

После создания шаблона исключений для табличного списка и добавления его в Knowledge Base вы сможете по ссылкам из сводок о корреляционных событиях, зарегистрированных в MaxPatrol SIEM, автоматически добавлять исключения в белый список или удалять исключения из черного списка.

Для создания шаблона исключений используется разметка YAML. Шаблон состоит из секции `criteria`, которая должна содержать хотя бы одну секцию `filter` для условия применения шаблона и одну секцию `exceptions` для описания исключений.

> **Примечание.** В шаблоне исключений вы можете использовать символ # для объявления однострочного комментария. Все последующие символы в строке игнорируются при использовании шаблона.

Структура шаблона исключений

```
criteria:
  - filter: <Условие шаблона 1>
    exceptions:
        <Ключ 11>
            ...
  - filter: <Условие шаблона 2>
    exceptions:
        <Ключ 21>
            ...
...
```

Секция filter

Секция предназначена для ввода условия применения шаблона исключений. В условии рекомендуется указывать системные названия правил корреляции, которые используют табличный список для хранения исключений.

Условие может состоять из одного или нескольких предикатов вида: `<Поле события><Оператор><Значение>`. В предикате можно использовать логические операторы равенства (`=`) и неравенства (`!=`) или оператор `in` для проверки на наличие значения поля события в списке. Для формирования списка с оператором `in` нужно использовать квадратные скобки. Между предикатами можно использовать операторы `and` или `or`. С оператором `or` также можно использовать круглые скобки.

Пример:

```
    - filter: >
        correlation_name in [
            "Bruteforce_attempt_atomic",
            "Bruteforce_attempt_by_user",
```

```
            "Bruteforce_attempt_from_src",
            "Bruteforce_attempt_to_dst",
            "Bruteforce_success",
            "Cisco_FW_Execute_show_run_after_success_bruteforce",
            "User_enumeration"
        ]
```

Примечание. При использовании шаблона каждый разрыв строки после символа > и до пустой строки заменяется пробелом.

Секция exceptions

Секция предназначена для ввода описаний исключений. Для каждого исключения создается отдельная секция с уникальным в рамках шаблона названием. Это название является идентификатором (ключом) для сопоставления описания исключения и текста исключения, отображаемого для корреляционных событий в веб-интерфейсе MaxPatrol SIEM.

Каждая секция с описанием исключения должна содержать секцию `insert` для ввода правил заполнения колонок белого списка или секцию `remove` для ввода условия удаления записи из черного списка. Также секция `insert` может содержать секцию `filter` для ввода дополнительного условия для исключения.

Формат ввода:

```
<Ключ 1>
    filter: <Условие исключения 1>
    insert:
        <Название колонки 1>: <Значение 11>
        <Название колонки 2>: <Значение 12>
        ...
<Ключ 2>
    filter: <Условие исключения 2>
    remove:
        <Название колонки 1>: <Значение 21>
        <Название колонки 2>: <Значение 22>
        ...
```

Секция `filter` предназначена для ввода дополнительного условия для исключения. При проверке соответствия данных события условию исключения это условие с помощью оператора `and` объединяется с условием шаблона. В условии рекомендуется указывать данные события, по которым создается исключение, например имя узла, на котором зарегистрировано событие, или идентификатор учетной записи, с которой связана подозрительная активность.

Условие может состоять из одного или нескольких предикатов вида: `<Поле события><Оператор><Значение>`. В предикате можно использовать логические операторы равенства (`=`) и неравенства (`!=`) или оператор `in` для проверки на наличие значения поля

события в списке. Для формирования списка с оператором `in` нужно использовать квадратные скобки. Между предикатами можно использовать операторы `and` или `or`. С оператором `or` также можно использовать круглые скобки.

Секция `insert` предназначена для ввода правил заполнения колонок белого списка данными из события при добавлении исключения. Для каждой колонки нужно создать отдельную секцию с названием колонки и указать в ней правило ее заполнения. В секции вы можете указать:

— Конкретное значение, соответствующее типу данных колонки (в кавычках). Формат ввода:

```
<Название колонки>:
    "<Значение>"
```

Примечание. Если данные в колонке могут принимать любые значения, необходимо ввести звездочку (*) для колонки с типом данных String и ноль для колонки с типом данных Number.

— Значение из поля события. Формат ввода:

```
<Название колонки>:
    field:<Название поля>
```

— Значения из нескольких полей события, объединенные в одну строку с разделителем. Формат ввода:

```
<Название колонки>:
    join:
        fields: [<Название поля 1>, <Название поля 2>]
        separator: "<Разделитель>"
```

— Значение из поля события в нижнем регистре. Формат ввода:

```
<Название колонки>:
    lower:
        field: <Название поля>
```

— Значение из поля события в верхнем регистре. Формат ввода:

```
<Название колонки>:
    upper:
        field: <Название поля>
```

Секция `remove` предназначена для ввода условия (по данным из события) удаления записей из черного списка. Для каждой колонки табличного списка нужно создать отдельную секцию с названием колонки и указать в ней условие на соответствия данных события значению в колонке. При удалении исключения указанные для каждой колонки условия объединяются с помощью оператора `and`. Первая запись, удовлетворяющая такому составному условию, удаляется из табличного списка. При составлении условия для колонки вы можете указывать конкретные значения или значения из полей события, используя указанный выше для секции `insert` формат ввода.

Пример:

```
exceptions:
    # Фильтрация по пользователю
    By_object_user:
        filter: object.name = "account"
```

```
    insert:
        host: "*"
        rule: "*"
        specific_value: "*"
        user_domain:
            lower:
                field: object.domain
        user_id:
            lower:
                field: object.id
        user_name:
            lower:
                field:  object.name
# Фильтрация по пути к объекту
By_object_path_name_in_specific_value_for_rule:
    filter: >
        correlation_name in [
            "Detect_Abusing_CredSSP",
            "Detect_Account_Discovery"
        ]
    insert:
        host: "*"
        rule:
            field: correlation_name
        specific_value:
            lower:
                join:
                    fields: [object.path, object.name]
                    separator: ""
        user_domain: "*"
        user_id: "*"
        user_name: "*"
```

8. Создание правила агрегации

В этом разделе описаны директивы и инструкции, используемые при создании правил агрегации, даны примеры и рекомендации.

Для отладки нового правила вы можете использовать утилиты SDK или утилиту PTSIEMSDK GUI. Добавить новое правило в MaxPatrol SIEM вы можете через веб-интерфейс Knowledge Base (см. Руководство оператора).

В этом разделе

8.1. Алгоритм работы службы агрегации

Из потока нормализованных и корреляционных событий служба агрегации отбирает те, которые подлежат агрегации. Эти события должны удовлетворять условию отбора, указанному при объявлении события хотя бы в одном из правил агрегации (в директиве `event`). Для каждого правила агрегации отобранные события разделяются на потоки событий с одинаковыми значениями всех полей, указанных при объявлении события (в инструкции `key` директив `event`).

Условие правила агрегации (в директиве `aggregate`) содержит описание последовательности событий, при появлении которой регистрируется агрегированное событие. При составлении одной последовательности используются события из одного потока, одновременно могут составляться несколько последовательностей для разных потоков. Каждое отобранное событие используется при составлении последовательности, указанной в условии правила агрегации.

При регистрации первого события в последовательности начинается отсчет времени регистрации всей последовательности и количества событий в последовательности.

Если время регистрации всей последовательности не превышено, выполняется подсчет количества событий в последовательности:

— Если номер события меньше числа пропускаемых событий, событие остается в потоке событий. Количество событий в последовательности увеличивается на единицу.

— Если номер события равен или больше числа пропускаемых событий, но меньше числа событий для регистрации агрегированного события, событие удаляется из потока событий. Количество событий в последовательности увеличивается на единицу.

— Если номер события равен числу событий для регистрации агрегированного события, событие удаляется из потока событий, регистрируется агрегированное событие.

Если время регистрации всей последовательности превышено, то в зависимости от количества событий в последовательности принимается решение о регистрации агрегированного события:

— Если количество событий меньше числа пропускаемых событий, агрегированное событие не регистрируется.

— Если количество событий больше числа пропускаемых событий, регистрируется агрегированное событие.

См. также

Этапы обработки событий (см. раздел 2)

Программные компоненты для обработки событий (см. раздел 3)

8.2. Структура правила агрегации

Правило агрегации должно состоять из директив `event` и `aggregate`.

Директива `event` используется для объявления события, подлежащего агрегации. Директива должна содержать название события, условие его отбора и может содержать инструкцию `key` для указания полей, по совокупности значений которых разделяется поток событий.

Директива `aggregate` используется для описания правила агрегации. Директива должна содержать уникальное название правила агрегации и условие его выполнения. В условии правила агрегации нужно указать, какое число раз и в течение какого времени объявленное событие должно появиться в потоке событий для регистрации агрегированного события.

> **Примечание.** Все поля агрегированного события заполняются автоматически. В событии указываются все поля, перечисленные в инструкции `key` и указанные в условии отбора события директивы `event`. В поле `aggregation_name` указывается название правила агрегации, по которому зарегистрировано событие, в поле `count` — количество событий в последовательности, в поле `start_time` — время регистрации первого события в последовательности.

Структура правила агрегации:

```
event <Название события>:
    key:
        <Поле события 1>, <Поле события 2>, ...
    <Условие отбора события>
aggregate <Название правила>: <Условие правила агрегации>
```

8.3. Директива event. Объявление события, подлежащего агрегации

Директива `event` предназначена для объявления события, подлежащего агрегации. Директива должна содержать название события (должно начинаться с прописной буквы латинского алфавита), условие его отбора из потока нормализованных (и корреляционных) событий и может содержать инструкцию `key` для указания полей события, по совокупности значений которых для каждого правила агрегации поток отобранных событий разделяется на несколько потоков.

Формат вызова:

```
event <Название события>:
    key:
        <Поле события 1>, <Поле события 2>, ...
    <Условие отбора события>
```

Условие отбора события

Условия отбора события может состоять из нескольких пар «поле события — значение», записанных каждая в отдельной строке. Для ввода значения поля события в паре используется присваивание (вместо логического равенства «==»). Все пары в условии объединяются операцией конъюнкции (логическое И). Если условие отбора возвращает True, событие используется при составлении последовательности в условии правила агрегации.

Инструкция key

В инструкции `key` вы можете в одной строке, через запятую указать поля события, по совокупности значений которых в каждом правиле агрегации поток отобранных событий разделяется на несколько потоков.

> **Примечание.** Если при разделении на несколько потоков значения двух полей события в инструкции `key` могут взаимозаменять друг друга, вы можете перед такими полями добавить восклицательный знак. Например, при записи `!src.host, !dst.host` в один поток попадут события, соответствующие потокам `src.host, dst.host` и `dst.host, src.host`.

Пример

```
event Host_info:
    key:
        event_src.host, event_src.hostname, event_src.ip, object, object.value,
object.group, object.domain, action, status, importance, event_src.category,
event_src.vendor, event_src.title
    msgid = "OTHER_EV_UPDATE_INFO_ABOUT_HOST"
    object.type = "host"
```

8.4. Директива aggregate. Условие правила агрегации

Директива `aggregate` предназначена для описания правила агрегации. Директива должна содержать название правила агрегации и условие его выполнения. Название правила агрегации должно быть уникальным в рамках всего набора правил агрегации (может состоять из букв латинского алфавита, цифр, знаков подчеркивания и точки, должно начинаться с прописной буквы). В условии правила агрегации нужно указать:

— название события (событие должно быть объявлено в директиве `event`);

— максимальное количество появлений этого события в последовательности, при котором событие будет пропускаться;

— общее количество появлений этого события в последовательности;

— максимальное время регистрации всей последовательности. Для ввода времени используйте оператор отсчета времени `within` и сокращения: `s` — секунды, `m` — минуты, `h` — часы, `d` — дни. Время регистрации всей последовательности рассчитывается как разница между значениями полей `time` последнего и первого событий последовательности.

Условие правила агрегации возвращает True, если регистрируются все события последовательности за время, не превышающее указанного в операторе `within`, иначе — False. При выполнении условия регистрируется агрегированное событие.

> **Примечание.** Вы можете использовать правило агрегации для удаления из потока событий всех событий, удовлетворяющих условию отбора, указанному в директиве `event`. Формат вызова: `aggregate <Название правила>: <Название события>[-1, <Любое число>]`.

Формат вызова:

```
aggregate <Название правила>: <Название события>[<Количество пропускаемых событий>,
<Общее количество событий>] within <Время последовательности>
```

Пример

```
# Нормализованное событие 1: {
#    "dst.host":"10.0.12.45",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
#    "tenant_id": "00000005-d160-0de3-f000-0000abbac07a",
#    "scope_id": "00000005-d160-0de3-f000-0000abbac07b"}
```

```
# Нормализованное событие 2: {
#    "dst.host":"10.0.12.45",
#    "dst.port":23,
#    "time":"2017-10-18T12:00:10Z",
#    ...}
# Нормализованное событие 3: {
#    "dst.host":"10.0.12.45",
#    "dst.port":23,
#    "time":"2017-10-18T12:00:20Z",
#    ...}
event Event:
     key:
          event_src.vendor
      dst.host = "10.0.12.45"
      dst.port = 23
aggregate Example: Event[0,2] within 1m
# Пропущено событие 1, поскольку не удовлетворяет условию отбора
# Агрегированное событие: {
#    "aggregation_name":"Example",
#    "count":2,
#    "dst.host":"10.0.12.45",
#    "dst.port":23,
#    "start_time":"2017-10-18T12:00:10Z",
#    "time":"2017-10-18T12:00:20Z",
#    ...}
```

9. Создание правила обогащения

В этом разделе описаны директивы и инструкции, используемые при создании правил обогащения, даны примеры и рекомендации.

Для отладки нового правила вы можете использовать утилиты SDK или утилиту PTSIEMSDK GUI. Добавить новое правило в MaxPatrol SIEM вы можете через веб-интерфейс Knowledge Base (см. Руководство оператора).

В этом разделе

9.1. Структура правила обогащения

Правило обогащения состоит из последовательности директив, содержащих инструкции. Для работы правила обогащения необходимо указать обязательные директивы и инструкции.

Директива `event` предназначена для объявления события для обогащения. В одном правиле можно использовать несколько директив `event` для объявления нескольких событий с разными названиями. Директива должна содержать инструкцию `filter` для настройки условия отбора события.

Директива `enrichment` предназначена для объявления правила обогащения. В директиве нужно указать название правила, уникальное в рамках всех правил обогащения.

Директива `enrich` (обработчик события) предназначена для выполнения инструкций при регистрации события, указанного в директиве. Директиву `enrich` нужно указать для каждого объявленного события. Директива может содержать инструкции:

- `enrich_fields` — для заполнения полей события значениями, указанными в блоке операторов инструкции или полученными из табличного списка; также в инструкции можно настроить регистрацию инцидента по событию;
- `insert_into` — для записи в табличный список значений полей события;
- `remove_from` — для удаления строк из табличного списка.

Правило обогащения может содержать директиву `query` для объявления запроса к табличному списку, данными из которого обогащаются события. В одном правиле можно объявить несколько запросов с разными названиями.

Структура правила обогащения:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
    <Условие запроса>
    }
event <Название события>:
    filter {
        <Условие отбора события>
    }
enrichment <Название правила обогащения>
    enrich <Название события>:
        enrich_fields {
            <Блок операторов для обогащения события>
            <Блок для настройки инцидента>
        }
        insert_into <Название табличного списка> {
            <Блок операторов insert_into>
        }
        remove_from <Название табличного списка> {
            <Условие для удаления>
        }
```

9.2. Директива event. Объявление события для обогащения

Директива `event` предназначена для объявления события для обогащения. В каждом правиле обогащения нужно объявить хотя бы одно событие.

В директиве нужно указать название события и условие отбора события. Название должно начинаться с прописной буквы латинского алфавита и быть уникальным в рамках одного правила обогащения. Для указания условия отбора используется инструкция `filter`.

Инструкция filter

Инструкция `filter` директивы `event` предназначена для настройки условия отбора события. Если условие возвращает значение True, выполняется обогащение события. При написании условия вы можете использовать:

- математические и логические операторы;
- поля события;
- литералы (с типом данных Bool, Null, Number, String);
- инлайн-запросы к табличным спискам (с префиксом `table::`);

— вызовы макросов (с префиксом `filter::`);

— функции языка XP, в том числе для выполнения запроса к табличному списку (см. раздел 4.9.2).

Формат вызова:

```
event <Название события>:
     filter {
          <Условие отбора события>
     }
```

См. также

Префикс table::, инлайн-запрос (см. раздел 4.9.1)

Создание макроса (см. раздел 6)

9.3. Директива enrichment. Название правила обогащения

Директива `enrichment` предназначена для объявления названия правила обогащения. Название должно быть уникальным в рамках всего набора названий правил обогащения (может состоять из букв латинского алфавита, цифр, знаков подчеркивания и точки, должно начинаться с прописной буквы).

Формат вызова:

```
enrichment <Название правила обогащения>
```

9.4. Директива enrich. Обработчик события

Директива `enrich` (обработчик события) предназначена для выполнения блока операторов при регистрации указанного события. Событие должно быть объявлено в директиве `event`. В блоке операторов директивы `enrich` вы можете использовать инструкцию `enrich_fields`, `insert_into` и `remove_from`.

Формат вызова:

```
enrich <Название события>:
    enrich_fields {
        <Блок операторов для обогащения события>
        <Блок для настройки инцидента>
    }
    insert_into <Название табличного списка> {
        <Блок операторов insert_into>
    }
    remove_from <Название табличного списка> {
        <Условие для удаления>
    }
```

В этом разделе

9.4.1. Инструкция enrich_fields. Обогащение события

Инструкция `enrich_fields` директивы `enrichment` предназначена для заполнения полей указанного события значениями из табличного списка. В блоке операторов инструкции `enrich_fields` вы можете использовать переменные (названия нужно начинать со знака доллара), поля указанного в инструкции события, функции и операторы языка XP или литералы с типом данных, соответствующим заполняемым полям.

Формат вызова:

```
enrich <Название события:
     enrich_fields {
        <Блок операторов для обогащения события>
        <Блок для настройки инцидента>
     }
```

Пример

```
# Нормализованное событие: {
#    "protocol":"17",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event AnythingWithProtocol:
     filter {
          protocol != null and protocol != ""
     }
enrichment Protocol_number_to_name
```

```
    enrich AnythingWithProtocol:
          enrich_fields {
               switch protocol
                    case "1" protocol = "ICMP" # Internet Control Message
                    case "2" protocol = "IGMP" # Internet Group Management
                    case "6" protocol = "TCP" # Transmission Control
                    case "17" protocol = "UDP" # User Datagram
               endswitch
          }
# Нормализованное событие после обогащения: {
#    "protocol":"UDP",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
```

9.4.2. Инструкция insert_into. Запись в табличный список

Внимание! Вы можете изменять только табличные списки, предназначенные для правил обогащения. Назначение табличного списка вы можете выбрать при его создании в MaxPatrol SIEM.

Инструкция `insert_into` используется для записи в табличный список. Для использования инструкции нужно указать:

— название табличного списка (должно начинаться с прописной буквы латинского алфавита);

— имена колонок (с префиксом `column::`), в которые нужно добавить значения;

— значения ячеек для добавления в табличный список (тип данных должен совпадать с типом данных колонки табличного списка). При вычислении значения вы можете использовать поля события, функции и операторы языка XP.

Примечание. Если указано значение для ключевой колонки табличного списка и оно есть в табличном списке — значения строки перезаписывается, если нет — добавляется новая строка в конец табличного списка.

При добавлении строки в табличный список нужно указать значения ячеек для всех колонок. Если значения для колонок не указаны, результат записи зависит от свойств колонки, настроенных при создании табличного списка: если колонка может содержать null — в добавляемые ячейки будет записано значение null, если нет — появится сообщение об ошибке.

Формат вызова:

```
enrichment <Название правила обогащения>
    enrich <Название события>:
        insert_into <Название табличного списка> {
            column::<Имя колонки 1> = <Значение для записи 1>
            column::<Имя колонки 2> = <Значение для записи 2>
            ...
        }
```

Пример

```
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие 1: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11",
#    "dst.port":24,
#    "status":"Open"}
# Нормализованное событие 2: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac078",
#    "dst.ip":"192.0.2.11",
#    "dst.port":25,
#    "status":"Close"}
event Event:
     filter {
          dst.ip == "192.0.2.11"
     }
enrichment Example
    enrich Event:
        insert_into Tabular_List_EnrichmentRule {
            column::IP = dst.ip
            column::Port = dst.port
            column::Status = status
        }
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":24, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":25, "Status":"Close"}]
```

9.4.3. Инструкция insert_into. Условная запись в табличный список

> **Внимание!** Вы можете изменять только табличные списки, предназначенные для правил обогащения. Назначение табличного списка вы можете выбрать при его создании в MaxPatrol SIEM.

Инструкция `insert_into` может использоваться для записи в табличный список при выполнении условия. Для использования инструкции нужно указать:

- название табличного списка (должно начинаться с прописной буквы латинского алфавита);
- условие записи в табличный список. Если условие возвращает `true`, выполняется запись в табличный список, если `false` — не выполняется. Условие может состоять из поля события с типом данных Bool или переменной, объявленной в инструкции `enrich_fields` и содержащей данные с типом Bool;
- имена колонок (с префиксом `column::`), в которые нужно добавить значения;
- значения ячеек для добавления в табличный список (тип данных должен совпадать с типом данных колонки табличного списка). При вычислении значения вы можете использовать поля события, функции и операторы языка XP.

  > **Примечание.** Если указано значение для ключевой колонки табличного списка и оно есть в табличном списке — значения строки перезаписывается, если нет — добавляется новая строка в конец табличного списка.

При добавлении строки в табличный список нужно указать значения ячеек для всех колонок. Если значения для колонок не указаны, результат записи зависит от свойств колонки, настроенных при создании табличного списка: если колонка может содержать null — в добавляемые ячейки будет записано значение null, если нет — появится сообщение об ошибке.

Формат вызова:

```
enrichment <Название правила обогащения>
    enrich <Название события>:
        insert_into <Название табличного списка> if <Условие записи> {
            column::<Имя колонки 1> = <Значение для записи 1>
            column::<Имя колонки 2> = <Значение для записи 2>
            ...
        }
```

Пример

```
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие 1: {
#    "event_src.vendor":"Best Company",
```

```
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11",
#    "dst.port":24,
#    "status":"Open"}
# Нормализованное событие 2: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac078",
#    "dst.ip":"192.0.2.11",
#    "dst.port":25,
#    "status":"Close"}
event Event:
     filter {
          dst.ip == "192.0.2.11"
     }
enrichment Example
    enrich Event:
        enrich_fields {
            $f_cond = bool(dst.ip == "192.0.2.11")
        }
        insert_into Tabular_List_EnrichmentRule if $f_cond {
            column::IP = dst.ip
            column::Port = dst.port
            column::Status = status
        }
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":24, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":25, "Status":"Close"}]
```

9.4.4. Инструкция insert_into. Арифметические операции при записи в табличный список

> **Внимание!** Вы можете изменять только табличные списки, предназначенные для правил обогащения. Назначение табличного списка вы можете выбрать при его создании в MaxPatrol SIEM.

Для выполнения арифметических операций при записи в табличный список в инструкции `insert_into` используются функции, указанные в таблице ниже. Для использования функций нужно указать:

— название табличного списка (должно начинаться с прописной буквы латинского алфавита);

— имя ключевой колонки (с префиксом `column::`) в указанном табличном списке;

Примечание. Какая из колонок является ключевой, определяется при создании табличного списка. В отличие от обычной колонки, все значения ключевой колонки уникальны в рамках одного табличного списка.

— значение ключевой колонки для выбора строки, в которую будут записаны значения;

Примечание. Если ключевая колонка не содержит указанного в инструкции значения, в табличный список добавляется строка с этим ключом.

— имя колонки (с префиксом `column::`), в ячейку которой требуется записать результат применения функции;

— значение для записи (с типом данных Number), к которому применяется функция.

Примечание. Если к ячейке табличного списка, содержащей максимальное значение, поддерживаемое для типа данных Number (см. приложение А), будет применена функция `insert_inc`, значение ячейки не изменится. Если к ячейке табличного списка, содержащей минимальное значение, будет применена функция `insert_dec`, значение ячейки также не изменится.

Формат вызова:

```
enrichment <Название правила обогащения>
    enrich <Название события>:
        insert_into <Название табличного списка> {
            <Имя ключевой колонки> = <Значение ключа>
            insert_min(column::<Имя колонки 1>, <Значение для записи 1>)
            insert_max(column::<Имя колонки 2>, <Значение для записи 2>)
            insert_inc(column::<Имя колонки 3>)
            insert_dec(column::<Имя колонки 4>)
        }
```

Таблица 80. Функции для выполнения арифметических операций при записи в табличный список в директиве insert_into

Функция	Арифметическая операция
`insert_min(column::<Имя колонки>, $f)`	Сравниваются два значения: указанное во втором аргументе функции и содержащееся в ячейке колонки, указанной в первом аргументе. В ячейку заносится меньшее из значений. Если ячейка содержит null или в табличный список добавляется новая строка, в ячейку заносится значение, указанное во втором аргументе функции
`insert_max(column::<Имя колонки>, $f)`	Сравниваются два значения: указанное во втором аргументе функции и содержащееся в ячейке колонки, указанной в первом аргументе. В ячейку заносится большее из значений. Если ячейка содержит null или в табличный список добавляется новая строка, в ячейку заносится значение, указанное во втором аргументе функции

Функция	Арифметическая операция
`insert_inc(column::<Имя колонки>)`	Значение ячейки в указанной колонке увеличивается на единицу. Если в табличный список добавляется новая строка, в ячейку заносится `1`
`insert_dec(column::<Имя колонки>)`	Значение ячейки в указанной колонке уменьшается на единицу. Если в табличный список добавляется новая строка, в ячейку заносится `–1`

Пример

```
# Табличный список "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "MinDuration":12, "MaxDuration":51, "Count":11},
#    {"IP":"192.0.2.11", "Port":22, "MinDuration":63, "MaxDuration":71, "Count":23},
#    {"IP":"192.0.2.11", "Port":23, "MinDuration":9, "MaxDuration":35, "Count":6}]
# Нормализованное событие: {
#    "dst.hostname":"192.0.2.11",
#    "dst.port":21,
#    "status":"Open",
#    "duration":55,
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
    filter {
        dst.ip == "192.0.2.11"
    }
enrichment Example
    enrich Event:
        insert_into Tabular_List_EnrichmentRule {
            column::Port = dst.port
            insert_min(column::MinDuration, duration)
              insert_min(column::MaxDuration, duration)
          insert_inc(column::Count)
        }
# Табличный список "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "MinDuration":12, "MaxDuration":55, "Count":12},
#    {"IP":"192.0.2.11", "Port":22, "MinDuration":63, "MaxDuration":71, "Count":23},
#    {"IP":"192.0.2.11", "Port":23, "MinDuration":9, "MaxDuration":35, "Count":6}]
```

9.4.5. Инструкция remove_from. Удаление строк из табличного списка

> **Внимание!** Вы можете изменять только табличные списки, предназначенные для правил обогащения. Назначение табличного списка вы можете выбрать при его создании в MaxPatrol SIEM.

Инструкция `remove_from` предназначена для удаления строк из табличного списка. При выполнении правила корреляции удаляются строки, удовлетворяющие указанному в инструкции условию. В инструкции нужно указать название табличного списка и условие удаления строк. В условии можно использовать переменные, имена колонок табличного списка (с префиксом `column::`) и логические операторы.

Формат вызова:

```
enrichment <Название правила обогащения>
    enrich <Название события>:
        remove_from <Название табличного списка> {
            <Условие для удаления>
        }
```

Пример

```
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11",
#    "dst.port":22}
event Event:
    filter {
        dst.ip == "192.0.2.11"
    }
enrichment Example
    enrich Event:
        remove_from Tabular_List_EnrichmentRule {
            column::Port > 22
        }
# Табличный список: "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"}]
# Нормализованное событие: {
#    "event_src.vendor":"Best Company",
```

```
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11",
#    "dst.port":22}
```

9.4.6. Команда drop

Команда `drop` используется в директиве `enrich` для удаления обогащаемого события. Событие может быть использовано при корреляции событий, но не будет сохранено в хранилище событий.

Формат вызова:

```
drop
```

Пример

```
event Event:
    filter {
        protocol = null or protocol = ""
    }
enrichment Example
    enrich Event:
        drop
```

9.5. Директива query. Объявление запроса к табличному списку

Директива `query` может использоваться для объявления запроса к табличному списку. В директиве нужно указать:

- название запроса, уникальное в рамках правила обогащения. В одном правиле обогащения можно объявить несколько запросов с разными названиями к одному или нескольким табличным спискам. Обращение к запросу выполняется по его названию;
- название табличного списка, к которому выполняется запрос (должно начинаться с прописной буквы латинского алфавита);
- аргументы запроса — переменные, используемые при составлении условия запроса. Значения переменных передаются в запрос при выполнении запроса. Запрос можно выполнить с помощью функций языка XP `exec_query` или `select_query_first`;
- условие, проверяемое при выполнении запроса. В условии запроса можно использовать значения переменных, переданные в аргументах запроса, имена колонок табличного списка (с префиксом `column::`), логические операторы и функции языка XP `in_subnet`, `regex_match`.

Примечание. При использовании функции `in_subnet` в условии запроса `query` к табличному списку в первом аргументе функции нужно указать имя переменной, переданной в запрос, во втором — имя колонки табличного списка (с префиксом `column::`).

Формат вызова:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
}
```

Пример

```
# Табличный список: "Process_PID": [
#    {"key":1, "Host":"192.0.2.11","Path":"c:\\windows\\cmd.exe", "PID":"32571",
"Last_Changed":1512099762, "ID":0}]
# Нормализованное событие: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "action": "open",
#    "object": "file",
#    "event_src.host":"192.0.2.11",
#    "datafield1": "32571"}
query GetProcessPath($host, $pid) from Process_PID {
    column::Host == $host and
    column::PID == $pid
}
event ModuleLoadedByProcess:
    filter {
        action == "open"
        and object == "file"
    }
enrichment Map_process_pid_to_path
enrich ModuleLoadedByProcess:
    enrich_fields {
        datafield4 = select_query_first("GetProcessPath", [event_src.host,
datafield1], "Path")
    }
# Нормализованное событие после обогащения: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "action": "open",
#    "object": "file",
#    "event_src.host":"192.0.2.11",
#    "datafield1": "32571",
#    "datafield4":"c:\\windows\\cmd.exe"}
```

В этом разделе

Инструкции qhandler, limit, skip (см. раздел 9.5.1)

Агрегатные функции для результатов запроса к табличному списку (см. раздел 9.5.2)

См. также

in_subnet (см. раздел 4.8.3)

Функция regex_match, проверка строки по регулярному выражению из табличного списка (см. раздел 4.9.5)

Функция exec_query, запрос к табличному списку (см. раздел 4.9.2)

Функция select_query_first, запрос значения из табличного списка (см. раздел 4.9.4)

9.5.1. Инструкции qhandler, limit, skip

Инструкция `qhandler` используется для выполнения арифметических операций над результатом запроса к табличному списку. При выполнении запроса с инструкцией `qhandler` функция `exec_query` возвращает ассоциативный массив (с типом данных KeyValue), содержащий результаты вычислений (с типом данных Number) для указанных в инструкции агрегатных функций. Для использования инструкции нужно указать:

— агрегатную функцию для выполнения арифметической операции;

— имя колонки табличного списка (с префиксом `column::`), элементам которого применяется агрегатная функция (поддерживаются операции над элементами колонок с типом данных Number);

— ключ для получения результата применения агрегатной функции из ассоциативного массива, возвращаемого функцией `exec_query`.

Инструкция `limit` используется для указания количества первых строк в результате запроса, к которым применяются агрегатные функции, указанные в инструкции `qhandler`.

Инструкция `skip` используется для исключения указанного количества первых строк из результата запроса, к оставшимися строками применяются агрегатные функции, указанные в инструкции `qhandler`.

> **Примечание.** При использовании инструкций `limit` и `skip` нужно учитывать, что сортировка строк в результате запроса не определена.

Формат вызова:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
     }
     qhandler {
          <Ключ 1> = <Агрегатная функция 1>(column::<Имя колонки 1>)
          <Ключ 2> = <Агрегатная функция 2>(column::<Имя колонки 2>)
          ...
```

```
    }
    limit <Максимальное число возвращаемых строк>
    skip <Количество строк, исключаемых из результата>
```

Пример

```
# Табличный список "Tabular_List_EnrichmentRule": [
#    {"IP":"192.0.2.10", "Port":22, "Status":"Open", "Duration":41},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":33},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":67}]
# Нормализованное событие: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11"}
# Запрос TestQuery к табличному списку Tabular_List_EnrichmentRule
query TestQuery ($check_ip) from Tabular_List_EnrichmentRule {
    column::IP == $check_ip and
    column::Port == 22 and
    column::Status == "Open"
}
    qhandler {
        $get_max_duration = max(column::Duration)
        $get_min_duration = min(column::Duration)
        $get_avg_duration = avg(column::Duration)
        $get_count = count()
    }
event Event:
    filter {
        event_src.vendor == "Best Company"
        }
enrichment Example
enrich Event:
   enrich_fields {
        # Присваивание переменным максимального, минимального и среднего времени
открытия порта 22 для переданного в запрос IP-адреса узла
        $get_duration = exec_query("TestQuery", [dst.ip])
        $max_duration = $get_duration["$get_max_duration"]
        $min_duration = $get_duration["$get_min_duration"]
        $avg_duration = $get_duration["$get_avg_duration"]
        $count = $get_duration["$get_count"]
        datafield1 = "IP_" + string(dst.ip)
        datafield2 = "Max_duration_" + string($max_duration)
        datafield3 = "Min_duration_" + string($min_duration)
        datafield4 = "Average_duration_" + string($avg_duration)
        datafield5 = "Result_count_" + string($count)
    }
```

```
# Нормализованное событие после обогащения: {
#    "event_src.vendor":"Best Company",
#    "time":"2017-10-18T12:00:00Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac077",
#    "dst.ip":"192.0.2.11",
#    "datafield1":"IP_192.0.2.11",
#    "datafield2":"Max_duration_67",
#    "datafield3":"Min_duration_33",
#    "datafield4":"Average_duration_50",
#    "datafield5":" Result_count_3"}
```

9.5.2. Агрегатные функции для результатов запроса к табличному списку

Указанные в таблице агрегатные функции используются только в инструкции `qhandler` для выполнения арифметических операций над результатами запроса к табличному списку.

Таблица 81. Агрегатные функции для выполнения арифметических операций над результатами запроса к табличному списку

Агрегатная функция	Возвращаемое значение
`min(column::<Имя колонки>)`	Наименьший элемент в результатах запроса
`max(column::<Имя колонки>)`	Наибольший элемент в результатах запроса
`avg(column::<Имя колонки>)`	Среднее значение элементов, округленное до целого
`median(column::<Имя колонки>)`	Значение медианного элемента. При четном числе элементов выбирается нижнее медианное значение
`sum(column::<Имя колонки>)`	Сумма всех элементов результатов запроса
`count()`	Количество элементов в результатах запроса
`column::<Имя колонки>`	Первый элемент из результатов запроса

10. Создание правила корреляции

В этом разделе описаны директивы и инструкции, используемые при создании правил корреляции, даны примеры и рекомендации.

Для отладки нового правила вы можете использовать утилиты SDK или утилиту PTSIEMSDK GUI. Добавить новое правило в MaxPatrol SIEM вы можете через веб-интерфейс Knowledge Base (см. Руководство оператора).

В этом разделе

10.1. Алгоритм работы службы корреляции

Из потока нормализованных событий служба маршрутизации отбирает те, которые подлежат корреляции. Эти события должны удовлетворять условию, указанному хотя бы в одном из правил корреляции (в инструкции `filter` директивы `event`).

Отобранные события поступают в службу корреляции событий, где для каждого правила корреляции разделяются на потоки в соответствии со значениями полей, указанными при объявлении событий (в инструкциях `key` директив `event`). Для каждого события выполняется проверка условия правила корреляции (указанного в директиве `rule`):

— если событие является первым в последовательности событий, указанной в условии правила корреляции, в памяти MP SIEM Server создается цепочка из одного события (выполняются инструкция `init` директивы `rule` и инструкция `on` для события);

Примечание. В памяти MP SIEM Server одновременно сохраняются все цепочки, составленные из различных комбинаций событий, поступивших в службу корреляции, и соответствующие последовательностям событий, указанным в условиях всех правил корреляции.

— если событие является не первым и не последним в последовательности — в цепочку добавляется следующее событие (для события выполняется инструкция `on`);

— если событие является последним в последовательности, цепочка событий совпадает с указанной в условии правила последовательностью и собрана за указанное в условии время — регистрируется корреляционное событие (для события выполняется инструкция `on` и директива `emit`), цепочка событий удаляется из памяти MP SIEM Server.

Цепочка событий может быть удалена из памяти MP SIEM Server без регистрации корреляционного события:

— если истекло время, указанное в операторе отсчета времени в условии правила корреляции;

— если истекло время между моментами регистрации событий, удовлетворяющих условию отбора для правила корреляции (по умолчанию — один час);

— если истекло время жизни цепочки событий (по умолчанию — сутки);

Примечание. Вы можете изменить время между моментами регистрации событий и время жизни цепочки событий в конфигурационном файле службы корреляции событий, используя параметры `mtime_ttl` и `ctime_ttl` соответственно.

— если из правила корреляции из инструкции `on` подана команда `close`.

См. также

Этапы обработки событий (см. раздел 2)

Программные компоненты для обработки событий (см. раздел 3)

10.2. Структура правила корреляции

Правило корреляции состоит из последовательности директив, содержащих инструкции и команды. Для работы правила корреляции необходимо указать обязательные директивы и инструкции.

Директива `event` используется для объявления события, подлежащего корреляции. Для объявления нескольких событий с разными названиями в одном правиле можно использовать несколько директив `event`. Директива может содержать инструкцию `key` для указания полей события, по значению которых разделяется поток событий, и должна содержать инструкцию `filter` для настройки условия отбора события.

Директива `rule` используется для описания правила корреляции. В директиве нужно указать условие выполнения правила корреляции. Название, указанное в директиве, является уникальным названием правила корреляции. Все события, используемые при составлении условия правила корреляции, должны быть объявлены с помощью директив `event`. Директива

может содержать инструкцию `init` для объявления переменных и по одной инструкции `on` для каждого объявленного события. Инструкция `on` (обработчик события) используется для выполнения блока операторов при регистрации события, указанного в этой инструкции. Директива `rule` может содержать инструкции `insert_into`, `remove_from`, `clear_table` для редактирования табличного списка.

Директива `emit` используется для заполнения полей корреляционного события и настройки регистрации инцидента (см. раздел 11) при выполнении правила корреляции.

Правило корреляции может содержать директиву `query` для объявления запроса к табличному списку. В одном правиле можно объявить несколько запросов с разными названиями.

Структура правила корреляции:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
     }
event <Название события>:
     key:
          <Поле события 1>, <Поле события 2>, ...
     filter {
          <Условие отбора события>
     }
rule <Название правила>: <Условие правила корреляции>
     init {
          <Блок операторов init>
     }
     on <Название события> {
          <Блок для заполнения полей события>
     }
     insert_into <Название табличного списка> {
          <Блок операторов insert_into>
     }
emit {
     <Блок для заполнения полей события>
     <Блок для настройки инцидента>
}
```

10.3. Заполнение полей корреляционного события

В правиле корреляции требуется указать значения обязательных для заполнения полей корреляционного события (см. раздел 10.3). Кроме того, вы можете указать значения необязательных полей или изменить значения отдельных сервисных полей, заполняемых автоматически. Заполнение полей корреляционного события выполняется в инструкциях `on` директивы `rule` и в директиве `emit` правила корреляции.

> **Внимание!** При заполнении полей корреляционного события их названия нужно начинать со знака доллара ($).

Нужно указать значения для следующих обязательных для заполнения полей корреляционного события:

— `correlation_type` — для настройки регистрации инцидента (см. раздел 11); если указано `event`, регистрируется только корреляционное событие, если `incident` — корреляционное событие и инцидент;

— `action`, `object`, `status` — для описания характера, объекта и результата воздействия; могут принимать только определенные значения (см. приложение В);

Примечание. Вы можете заполнить необязательное поле `subject`, чтобы указать источник воздействия.

— `category.generic`, `category.high`, `category.low` — для назначения категории нарушения информационной безопасности, в связи с которым регистрируется корреляционное событие; рекомендуется заполнять определенными значениями (см. приложение Д);

— `importance` — для указания важности события для информационной безопасности; может принимать значения `info`, `low`, `medium`, `high`;

— `id` — для ввода идентификатора правила корреляции в виде: `<Пространство имен>_SIEM_<Название правила корреляции>`.

10.4. Директива event. Объявление события, подлежащего корреляции

Директива `event` предназначена для объявления события, подлежащего корреляции. В каждом правиле корреляции нужно объявить хотя бы одно событие.

В директиве нужно указать название события, поля события для разделения потока нормализованных событий и условие отбора события. Название должно начинаться с прописной буквы латинского алфавита и быть уникальным в рамках одного правила корреляции. Для указания полей события и условия отбора используются инструкции `key` и `filter`.

Формат вызова:

```
event <Название события>:
     key:
          <Поле события 1>, <Поле события 2>, ...
     filter {
          <Условие отбора события>
     }
```

Пример

```
# Нормализованное событие 1: {
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
```

```
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Нормализованное событие 2: {
#    "dst.port":23,
#    "time":"2017-10-18T12:00:10Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac076",
#    ...}
# Нормализованное событие 3: {
#    "dst.port":23,
#    "time":"2017-10-18T12:00:20Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac075",
#    ...}
event Event_1:
     key:
          event_src.vendor
     filter {
          dst.port == 22
     }
event Event_2:
     key:
          event_src.vendor
     filter {
          dst.port == 23
     }
rule Example: Event_1 -> Event_2[2]
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ...}
```

В этом разделе

10.4.1. Инструкция key

Инструкция `key` директивы `event` предназначена для указания полей события, по значению которых поток нормализованных событий будет разделяться на несколько потоков. В инструкции вы можете указать одно поле события или несколько полей через запятую. Разделение потока событий выполняется по совокупности значений указанных полей. Наборы полей для событий, объявленных в одном правиле корреляции, могут отличатся названиями полей, но типы данных для полей в одинаковых позициях должны совпадать.

> **Примечание.** Вы можете не указывать инструкцию `key`, если условие правила корреляции выполняется при регистрации одного события (не указана последовательность событий).

Пример:

```
event Account_login_failed:
    key:
        event_src.host, subject.name
    filter {
        ...
    }
event Account_login_failed_dst:
    key:
        dst.host, subject.name
    filter {
        ...
    }
```

10.4.2. Инструкция filter

Инструкция `filter` директивы `event` предназначена для настройки условия отбора события. Если условие возвращает значение True, событие используется в правиле корреляции. При написании условия вы можете использовать:

- математические и логические операторы;
- поля события;
- литералы (с типом данных Bool, Null, Number, String);
- инлайн-запросы к табличным спискам (с префиксом `table::`);
- вызовы макросов (с префиксом `filter::`);
- функции языка XP, в том числе для выполнения запроса к табличному списку (см. раздел 4.9.2).

Упрощенный вызов инструкции filter

Вы можете опустить название инструкции `filter` и операторы `and`, если в инструкции `filter` в условии используется только конъюнкция (логическое И) и указаны точные значения полей события. Для ввода значений полей в этом случае используется присваивание (вместо логического равенства `==`).

Пример:

```
event SessionStop:
     key: src.ip
          action = "stop"
          object = "session"
          status = "success"
          subject.domain = "net"
```

Многоуровневая корреляция

Многоуровневая корреляция — поиск событий информационной безопасности на основе анализа зарегистрированных ранее корреляционных событий.

Для отбора корреляционных событий в инструкции `filter` для поля `correlation_name` нужно указать название правила корреляции, по которому были зарегистрированы корреляционные события.

Пример:

```
event Event:
     key:
          correlation_name
     filter {
          correlation_name == "Cisco_WebVPN_start_success"
     }
```

Корреляция на активах

Корреляция на активах — поиск событий информационной безопасности, связанных с активами, зарегистрированными в MaxPatrol SIEM.

Для отбора связанных с активом событий нужно в инструкции `filter` для одного из полей `src.asset`, `dst.asset`, `event_src.asset` или `recv_asset` указать UUID актива.

Пример:

```
event Asset_event:
     key:
          event_src.asset
     filter {
          event_src.asset != null and
          action == "login" and
```

```
        status == "success"
    }
```

См. также

Префикс table::, инлайн-запрос (см. раздел 4.9.1)

Создание макроса (см. раздел 6)

10.5. Директива rule. Условие правила корреляции

Директива `rule` предназначена для описания правила корреляции. В директиве нужно указать название правила корреляции и условие его выполнения. Название должно быть уникальным в рамках всего набора правил корреляции (может состоять из букв латинского алфавита, цифр, знаков подчеркивания и точки, должно начинаться с прописной буквы).

Условие правила корреляции может содержать названия событий, объявленных в директивах `event`, и специальные операторы для описания последовательности событий.

Последовательность может содержать одно или несколько событий. При описании последовательности событий вы можете указать порядок регистрации событий, количество появлений каждого события, интервалы времени между моментами регистрации событий или ограничить время регистрации всей последовательности.

Если регистрируются все события последовательности в указанном порядке и за указанное время, условие правила корреляции возвращает True, иначе — False. При выполнении условия правила корреляции выполняется блок операторов директивы `emit` и регистрируется корреляционное событие.

В директиве `rule` можно использовать инструкции `init` и `on`. Если регистрируется первое (крайнее слева) событие последовательности, указанной в условии, выполняются блок операторов инструкции `init` и блок операторов инструкции `on` для зарегистрированного события. При регистрации следующих событий последовательности для них выполняются блоки операторов в инструкциях `on`.

> **Примечание.** В директиве `rule` для редактирования табличного списка вы можете использовать инструкции `insert_into`, `remove_table` и `clear_table`. Инструкции выполняются при выполнении условия правила корреляции.

Формат вызова:

```
rule <Название правила>: <Условие правила корреляции>
     init {
          <Блок операторов init>
     }
     on <Название события> {
          <Блок операторов on>
     }
```

Пример

```
# Нормализованное событие 1: {
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Нормализованное событие 2: {
#    "dst.port":23,
#    "time":"2017-10-18T12:00:10Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac076",
#    ...}
# Нормализованное событие 3: {
#    "dst.port":23,
#    "time":"2017-10-18T12:00:20Z",
#    "uuid":"00000005-9f0a-0e90-f000-0000abbac075",
#    ...}
event Event_1:
     key:
          event_src.vendor
     filter {
          dst.port == 22
     }
event Event_2:
     key:
          event_src.vendor
     filter {
          dst.port == 23
     }
rule Example: Event_1 -> Event_2[2]
     init {
          $f1 = 3
          $f2 = 0
          $f3 = 0
     }
     on Event_1 {
          $f1= $f1+1
          $f2= $f2+1
     }
```

```
      on Event_2 {
           $f1= $f1+1
           $f3= $f3+1
      }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
     $datafield1 = $f1
     $datafield2 = $f2
     $datafield3 = $f3
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":6,
#    "datafield2":1,
#    "datafield3":2,
#    ...}
```

В этом разделе

Логические операторы для последовательности событий (см. раздел 10.5.1)

Квантификаторы событий последовательности (см. раздел 10.5.2)

Операторы отсчета времени для последовательности (см. раздел 10.5.3)

Оператор as. Псевдонимы события (см. раздел 10.5.4)

Инструкция init. Объявление переменных (см. раздел 10.5.5)

Инструкция on. Обработчик события (см. раздел 10.5.6)

Команда close (см. раздел 10.5.7)

10.5.1. Логические операторы для последовательности событий

При составлении в условии правила корреляции последовательности из объявленных событий вы можете использовать логические операторы `->`, `or`, `and`, `not`, `with different`.

Оператор «->»

Результатом применения оператора `->` является True, если зарегистрировано событие, указанное слева от оператора, а затем событие, указанное справа от оператора. Вы можете использовать оператор для составления последовательности любой длины.

Формат вызова:

```
rule <Название правила>: <Название события 1> -> <Название события 2>
```

Пример:

```
rule Example: Event_1 -> Event_2
# Выполнено при регистрации: Event_1, Event_2.
```

Оператор «or»

Результатом применения оператора `or` является True, если зарегистрировано событие, указанное слева от оператора, или событие, указанное справа от оператора. Вы можете использовать оператор для составления последовательности любой длины.

Формат вызова:

```
rule <Название правила>: <Название события 1> or <Название события 2>
```

Пример:

```
rule Example: Event_1 or Event_2
# Выполнено при регистрации: Event_1 или Event_2.
```

Оператор «and»

Результатом применения оператора `and` является True, если зарегистрированы оба указанных события (в любом порядке). Вы можете использовать оператор для составления последовательности любой длины.

Формат вызова:

```
rule <Название правила>: <Название события 1> and <Название события 2>
```

Пример:

```
rule Example: Event_1 and Event_2
# Выполнено при регистрации: Event_1, Event_2 или Event_2, Event_1.
```

Оператор «not»

Результатом применения оператора `not` является False при регистрации события, указанного справа от оператора. Оператор нельзя использовать с первым или последним событием в последовательности.

> **Примечание.** Для использования оператора `not` с последним событием в последовательности нужно применить оператор отсчета времени `timer`.

Формат вызова:

```
rule <Название правила>: <Последовательность событий 1> -> not <Название события> ->
<Последовательность событий 2>
```

Пример:

```
rule Example: Event_1 -> not Event_2 -> Event_3
# Выполнено при регистрации: Event_1, Event_3.
# Не выполнено при регистрации: Event_1, Event_2.
```

Оператор «with different»

Результатом применения оператора `with different` является True, если во всех событиях последовательности, указанной слева от оператора, отличаются значения поля, указанного справа от оператора. Не поддерживается вложение операторов `with different` друг в друга.

> **Внимание!** Условие с оператором `with different` невыполнимо, если поле события в аргументе оператора совпадает с одним из полей, указанных в инструкции `key` директивы `event` для события, расположенного слева от оператора.

Формат вызова:

```
rule <Название правила>: (<Последовательность событий>) with different <Поле события>
```

Пример:

```
rule Example: (Event_1 -> Event_2) with different src.host
# Выполнено при регистрации: Event_1, Event_2 (с разными значениями поля src.host).
```

Пример:

```
rule Example: Event[3] with different(src.ip)
# Выполнено при регистрации: Event, Event, Event (с разными значениями поля src.ip).
```

Пример:

```
rule Example: (Event_1[2, 3] with different(src.ip)) -> Event_2
# Выполнено при регистрации: Event_1, Event_1 (с разными значениями поля src.ip),
Event_2 или Event_1, Event_1, Event_1 (с разными значениями поля src.ip), Event_2
```

Пример:

```
event Event:
     key:
          src.ip
     filter {
          event_src.vendor == "Best Company"
     }
rule Event[3] with different src.ip
# Всегда не выполнено
```

10.5.2. Квантификаторы событий последовательности

Квантификатор — оператор, применяемый к событию для указания количества его появлений подряд в последовательности событий. При составлении в условии правила корреляции последовательности из объявленных событий вы можете применять к событиям квантификаторы.

> **Примечание.** Если количество событий в последовательности не определено (последовательность оканчивается одним из операторов `?`, `*`, `+`, `[N,]`) и не указан оператор отсчета времени, собранная цепочка событий удаляется из памяти MP SIEM Server (без регистрации корреляционного события) по истечении времени жизни или по истечении времени между моментами регистрации событий (см. раздел 10.1).

Название события в условии

Результатом регистрации указанного события является True.

Формат вызова:

```
rule <Название правила>: <Название события>
```

Пример:

```
rule Example: Event
# Выполнено при регистрации: Event.
```

Оператор «+»

Результатом применения оператора `+` является True, если событие, указанное слева от оператора, зарегистрировано один раз или более. Оператор нельзя использовать для указания единственного или последнего события в последовательности с оператором отсчета времени `within`.

Формат вызова:

```
rule <Название правила>: <Название события>+
```

Пример:

```
rule Example: Event_1+ -> Event_2
# Выполнено при регистрации: Event_1, ..., Event_1, Event_2.
```

Оператор «?»

Результатом применения оператора `?` всегда является True. Вы можете использовать оператор для указания одного необязательного события в последовательности. Оператор нельзя использовать для указания первого или единственного события в любой последовательности и для указания последнего события в последовательности с оператором отсчета времени `within`.

Формат вызова:

```
rule <Название правила>: <Название события 1> -> <Название события 2>?
```

Пример:

```
rule Example: Event_1 -> Event_2? -> Event_1
# Выполнено при регистрации: Event_1, Event_1 или Event_1, Event_2, Event_1.
```

Оператор «*»

Результатом применения оператора * всегда является True. Вы можете использовать оператор для указания последовательности любой длины из одного необязательного события. Оператор нельзя использовать для указания первого или единственного события в любой последовательности и для указания последнего события в последовательности с оператором отсчета времени `within`.

Формат вызова:

```
rule <Название правила>: <Название события 1> -> <Название события 2>*
```

Пример:

```
rule Example: Event_1 -> Event_2* -> Event_1
# Выполнено при регистрации: Event_1, Event_1 или Event_1, Event_2 ,..., Event_1.
```

Оператор «[N,M]»

Результатом применения оператора `[N,M]` является True, если указанное слева от оператора событие зарегистрировано указанное число раз (см. таблицу 82).

Формат вызова:

```
rule <Название правила>: <Название события 1> -> <Название события 2>[N,M] ->
<Название события 3>
```

Таблица 82. Варианты настройки оператора [N,M]

Синтаксис	Число событий	Пример последовательности	Примечание
`<Название события>[N]`	Ровно N	`Event[3]` — последовательность из 3 событий	—
`<Название события>[N,] ...`	N и больше	`Event[3,]` — последовательность из 3, 4, 5, 6 и так далее событий	Вариант настройки нельзя использовать для указания последнего события в последовательности с оператором отсчета времени `within`

Синтаксис	Число событий	Пример последовательности	Примечание
`<Название события>[,M]`	M и меньше или отсутствует	`Event[,3]` — последовательность из 1, 2 или 3 событий или отсутствие события	Вариант настройки нельзя использовать для указания первого или единственного события в любой последовательности и для указания последнего события в последовательности с оператором отсчета времени `within`
`<Название события>[N,M]`	От N до M	`Event[3,5]` — последовательность из 3, 4 или 5 событий	Вариант настройки нельзя использовать для указания последнего события в последовательности с оператором отсчета времени `within`

Пример:

```
rule Example: Event[3]
# Выполнено при регистрации: Event, Event, Event.
```

Пример:

```
rule Example: Event_1[3,] -> Event_2
# Выполнено при регистрации: Event_1, Event_1, Event_1, Event_2 или Event_1, ..., Event_1, Event_2
```

Пример:

```
rule Example: Event_1 -> Event_2[,3] -> Event_1
# Выполнено при регистрации: Event_1, Event_1 или Event_1, Event_2, Event_1 или Event_1, Event_2, Event_2, Event_1 или Event_1, Event_2, Event_2, Event_2, Event_1.
```

Пример:

```
rule Example: Event [3, 5]
# Выполнено при регистрации: Event, Event, Event, или Event, Event, Event, Event или Event, Event, Event, Event, Event.
```

10.5.3. Операторы отсчета времени для последовательности

Внимание! В одном условии нельзя использовать несколько операторов отсчета времени.

При составлении в условии правила корреляции последовательности из объявленных событий вы можете использовать операторы отсчета времени.

Оператор within

Результатом применения оператора `within` является значение True, если разница между значениями полей `time` последнего и первого событий последовательности, указанной слева от оператора, меньше времени, указанного справа от оператора. Корреляционное событие регистрируется сразу при выполнении условия с оператором `within`.

Вы можете использовать оператор для событий последовательности для ограничения интервала времени между моментами их регистрации на источнике. Для ввода времени вы можете использовать сокращения: `s` — секунды, `m` — минуты, `h` — часы, `d` — дни. Последнее событие последовательности должно быть указано явно, для него нельзя использовать операторы `?`, `*`, `+`, `[N,]`, `[, M]`, `[N, M]`.

Формат вызова:

```
rule <Название правила>: <Последовательность событий> within <Время>
```

Пример:

```
rule Example: Event[3] within 5s
# Выполнено при регистрации не более чем за 5 секунд: Event, Event, Event.
```

Оператор timer

Результатом применения оператора `timer` является значение True, если все события последовательности, указанной слева от оператора, поступили в службу корреляции событий за указанное справа от оператора время.

Отсчет времени начинается с момента получения службой корреляции первого события последовательности и продолжается до истечения указанного времени. Корреляционное событие регистрируется после окончания указанного времени.

Вы можете использовать оператор для ограничения времени регистрации последовательности событий. Для ввода времени вы можете использовать сокращения: `s` — секунды, `m` — минуты, `1h` — 1 час.

Формат вызова:

```
rule <Название правила>: <Последовательность событий> timer <Время>
```

Пример:

```
rule Example: Event+ timer 5s
# Выполнено при регистрации за 5 секунд любого числа Event.
```

Пример:

```
rule Example: (Event_1 -> Event_2) timer 10m
# Выполнено при регистрации за 10 минуту Event_1, Event_2.
```

Пример:

```
rule Example: ((Event_1 -> Event_2?) timer 1h) -> Event_3
# Выполнено при регистрации за 1 час Event_1 или Event_1, Event_2, после этого за
любое время Event_3.
```

Оператор timeout_timer

Результатом применения оператора `timeout_timer` является значение True, если все события последовательности, указанной слева от оператора, поступают в службу корреляции событий с интервалом, не превышающим время, указанное справа от оператора.

Отсчет времени начинается с момента получения службой корреляции первого события последовательности и продолжается до истечения указанного времени. Если за это время в службу корреляции поступает следующее событие последовательности, отсчет времени перезапускается.

Вы можете использовать оператор для ограничения интервала времени между моментами регистрации событий последовательности. Для ввода времени вы можете использовать сокращения: `s` — секунды, `m` — минуты, `1h` — 1 час.

Формат вызова:

```
rule <Название правила>: <Последовательность событий> timeout_timer <Время>
```

Пример:

```
rule Example: Event[3] timeout_timer 10m
# Выполнено при регистрации за 30 минут: Event в течение первых 10 минут, затем Event
в течение следующих 10 минут, затем Event в течение последних 10 минут.
```

10.5.4. Оператор as. Псевдонимы события

Оператор `as` предназначен для объявления в условии правила корреляции директивы `rule` псевдонима объявленного события. Псевдонимы используются для выполнения блоков операторов в разных инструкциях `on` для одного события, указанного в разных частях последовательности.

Формат вызова:

```
rule <Название правила>: <Название события> as <Псевдоним>
     on <Псевдоним> {
          <Блок операторов on>
     }
```

Пример

```
rule Example: Event as Event_1 -> Event as Event_2
     on Event_1 {
          $f1=1
     }
     on Event_2 {
          $f1=2
     }
```

10.5.5. Инструкция init. Объявление переменных

Инструкция `init` директивы `rule` предназначена для объявления переменных, используемых в инструкциях `on` и `insert_into` или в директиве `emit`. Переменным вы можете присвоить определенные значения. Блок операторов инструкции `init` выполняется один раз при регистрации первого события последовательности в условии правила корреляции.

Формат вызова:

```
init {
     <Блок операторов init>
}
```

Пример

```
     init {
          $count.subevents = 0
     }
```

10.5.6. Инструкция on. Обработчик события

Инструкция `on` директивы `rule` (обработчик события) предназначена для выполнения блока операторов при регистрации указанного события (событие должно быть объявлено в директиве `event`).

В блоке операторов инструкции `on` вы можете объявлять переменные и присваивать значения полям корреляционного события. Для этого вы можете использовать поля указанного в инструкции события; переменные, объявленные ранее в инструкциях `init` и `on` (для других событий); функции и операторы языка XP.

Кроме этого, в блоке операторов инструкции `on` вы можете использовать команду `close` для удаления уже собранной цепочки событий из памяти MP SIEM Server без регистрации корреляционного события.

Формат вызова:

```
rule <Название правила>: <Условие правила корреляции>
     init {
          <Блок операторов init>
     }
     on <Название события> {
          <Блок операторов on>
     }
```

Пример

```
rule Bruteforce_success_by_user: (Login_failure[5,] -> Login_success) within 5m
     init{
          $count.subevents = 0
```

```
}
on Login_failure {
     $count.subevents = $count.subevents + 1
}
on Login_success {
     $subject.name = subject.name
     $subject.domain = subject.domain
     $count.subevents = $count.subevents + 1
}
```

10.5.7. Команда close

Команда `close` используется в инструкции `on` директивы `rule` для прекращения проверки последовательности событий и удаления уже собранной цепочки событий из памяти MP SIEM Server без регистрации корреляционного события.

Формат вызова:

```
close
```

Пример

```
rule Example: Event_1 -> Event_2
      on Event_1 {
            $hostname = src.hostname
      }
      on Event_2 {
            if $hostname != src.hostname then
                  close
            endif
      }
```

10.6. Директива rule. Изменение табличного списка

> **Внимание!** Из директивы `rule` вы можете изменять только табличные списки, предназначенные для правил корреляции. Назначение табличного списка вы можете выбрать при его создании в MaxPatrol SIEM.

В директиве `rule` для изменения табличного списка можно использовать инструкции `insert_into`, `remove_from` и `clear_table`. Инструкции выполняются при выполнении условия правила корреляции.

В этом разделе

Инструкция insert_into. Запись в табличный список (см. раздел 10.6.1)

Инструкция insert_into. Условная запись в табличный список (см. раздел 10.6.2)

10.6.1. Инструкция insert_into. Запись в табличный список

Инструкция `insert_into` может использоваться для записи в существующую строку (ячейку) табличного списка. Для использования инструкции нужно указать:

— название табличного списка (должно начинаться с прописной буквы латинского алфавита);

— переменные, в которых из директивы `rule` в инструкцию `insert_into` передаются значения ячеек табличного списка;

— имя ключевой колонки (с префиксом `column::`) в указанном табличном списке;

Примечание. Какая из колонок является ключевой, определяется при создании табличного списка. В отличие от обычной колонки, все значения ключевой колонки уникальны в рамках одного табличного списка.

— значение ключа для выбора строки, в которую будут записаны значения;

Примечание. Если ключевая колонка не содержит указанного в инструкции значения, в табличный список добавляется строка с этим ключом.

— имена колонок (с префиксом `column::`), в ячейки которых требуется записать значения;

— значения, которые будут записаны в ячейки табличного списка (тип данных должен совпадать с типом данных колонки табличного списка).

Формат вызова:

```
rule <Название правила>: <Условие правила>
     on <Название события> {
          $f_key = <Значение ключа>
          $f1 = <Значение 1>
          $f2 = <Значение 2>
          ...
     }
     insert_into <Название табличного списка> {
          column::<Имя ключевой колонки> = $f_key
          column::<Имя колонки 1> = $f1
          column::<Имя колонки 2> = $f2
          ...
     }
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":21,
#    "status":"Open",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
     key:
          event_src.vendor
     filter {
          dst.ip == "192.0.2.11"
     }
rule Example: Event
     on Event {
          $f_key = dst.port
          $f = status
     }
     insert_into Tabular_List_CorrelationRule {
          column::Port = $f_key
          column::Status = $f
}
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
```

```
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ...}
```

10.6.2. Инструкция insert_into. Условная запись в табличный список

Инструкция `insert_into` может использоваться для записи в табличный список при выполнении условия. Для использования инструкции нужно указать:

— название табличного списка (должно начинаться с прописной буквы латинского алфавита);

— условие записи в табличный список. Если условие возвращает `true`, выполняется запись в табличный список, если `false` — не выполняется. Условие может состоять из переменной, объявленной в директиве `rule` и содержащей данные с типом Bool;

— переменные, в которых из директивы `rule` в инструкцию `insert_into` передаются значения ячеек табличного списка;

— имя ключевой колонки (с префиксом `column::`) в указанном табличном списке;

Примечание. Какая из колонок является ключевой, определяется при создании табличного списка. В отличие от обычной колонки, все значения ключевой колонки уникальны в рамках одного табличного списка.

— значение ключа для выбора строки, в которую будут записаны значения;

Примечание. Если ключевая колонка не содержит указанного в инструкции значения, в табличный список добавляется строка с этим ключом.

— имена колонок (с префиксом `column::`), в ячейки которых требуется записать значения;

— значения, которые будут записаны в ячейки табличного списка (тип данных должен совпадать с типом данных колонки табличного списка).

Формат вызова:

```
rule <Название правила>: <Условие правила>
     on <Название события> {
          $f_key = <Значение ключа>
          $f1 = <Значение 1>
          $f2 = <Значение 2>
          ...
     }
     insert_into <Название табличного списка> if <Условие записи> {
          column::<Имя ключевой колонки> = $f_key
```

```
            column::<Имя колонки 1> = $f1
            column::<Имя колонки 2> = $f2
            ...
      }
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":21,
#    "status":"Open",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
      key:
            event_src.vendor
      filter {
            dst.ip == "192.0.2.11"
      }
rule Example: Event
      on Event {
            $f_cond = bool(dst.ip == "192.0.2.11")
            $f_key = dst.port
            $f = status
      }
      insert_into Tabular_List_CorrelationRule if $f_cond {
            column::Port = $f_key
            column::Status = $f
}
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
```

```
    $importance = "info"
}
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ...}
```

10.6.3. Инструкция insert_into. Арифметические операции при записи в табличный список

Для выполнения арифметических операций при записи в табличный список в инструкции `insert_into` используются функции, указанные в таблице ниже. Для использования функций нужно указать:

- название табличного списка (должно начинаться с прописной буквы латинского алфавита);
- переменные, в которых из директивы `rule` в инструкцию `insert_into` передаются значения ячеек табличного списка;
- имя ключевой колонки (с префиксом `column::`) в указанном табличном списке;

  Примечание. Какая из колонок является ключевой, определяется при создании табличного списка. В отличие от обычной колонки, все значения ключевой колонки уникальны в рамках одного табличного списка.

- значение ключа для выбора строки, в которую будут записаны значения;

  Примечание. Если ключевая колонка не содержит указанного в инструкции значения, в табличный список добавляется строка с этим ключом.

- имя колонки (с префиксом `column::`), в ячейку которой требуется записать результат применения функции;
- значение для записи (с типом данных Number), к которому применяется функция.

  Примечание. Если к ячейке табличного списка, содержащей максимальное значение, поддерживаемое для типа данных Number (см. приложение А), будет применена функция `insert_inc`, значение ячейки не изменится. Если к ячейке табличного списка, содержащей минимальное значение, будет применена функция `insert_dec`, значение ячейки также не изменится.

Формат вызова:

```
rule <Название правила>: <Условие правила>
     on <Название события> {
          $f_key = <Значение ключа>
          $f = <Значение для записи>
     }
     insert_into <Название табличного списка> {
          column::<Имя ключевой колонки> = $f_key
          insert_min(column::<Имя колонки 1>, $f)
          insert_max(column::<Имя колонки 2>, $f)
          insert_inc(column::<Имя колонки 3>)
          insert_dec(column::<Имя колонки 4>)
     }
```

Таблица 83. Функции для выполнения арифметических операций при записи в табличный список в директиве insert_into

Функция	Арифметическая операция
`insert_min(column::<Имя колонки>, $f)`	Сравниваются два значения: указанное во втором аргументе функции и содержащееся в ячейке колонки, указанной в первом аргументе. В ячейку заносится меньшее из значений. Если ячейка содержит null или в табличный список добавляется новая строка, в ячейку заносится значение, указанное во втором аргументе функции
`insert_max(column::<Имя колонки>, $f)`	Сравниваются два значения: указанное во втором аргументе функции и содержащееся в ячейке колонки, указанной в первом аргументе. В ячейку заносится большее из значений. Если ячейка содержит null или в табличный список добавляется новая строка, в ячейку заносится значение, указанное во втором аргументе функции
`insert_inc(column::<Имя колонки>)`	Значение ячейки в указанной колонке увеличивается на единицу. Если в табличный список добавляется новая строка, в ячейку заносится `1`
`insert_dec(column::<Имя колонки>)`	Значение ячейки в указанной колонке уменьшается на единицу. Если в табличный список добавляется новая строка, в ячейку заносится `-1`

Пример

```
# Табличный список "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "MinDuration":12, "MaxDuration":51, "Count":11},
#    {"IP":"192.0.2.11", "Port":22, "MinDuration":63, "MaxDuration":71, "Count":23},
#    {"IP":"192.0.2.11", "Port":23, "MinDuration":9, "MaxDuration":35, "Count":6}]
#Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
```

```
#    "dst.port":21,
#    "status":"Open",
#    "duration":55,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
     key:
          event_src.vendor
     filter {
          dst.ip == "192.0.2.11"
     }
rule Example: Event
     on Event {
          $f_key = dst.port
          $f = duration
     }
     insert_into Tabular_List_CorrelationRule {
          column::Port = $f_key
          insert_min(column::MinDuration, $f)
          insert_max(column::MaxDuration, $f)
          insert_inc(column::Count)
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Табличный список "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "MinDuration":12, "MaxDuration":55, "Count":12},
#    {"IP":"192.0.2.11", "Port":22, "MinDuration":63, "MaxDuration":71, "Count":23},
#    {"IP":"192.0.2.11", "Port":23, "MinDuration":9, "MaxDuration":35, "Count":6}]
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
```

```
#    "importance":"info",
#    ...}
```

10.6.4. Инструкция insert_into. Добавление строк в табличный список

Инструкция `insert_into` может использоваться для добавления одной или нескольких строк в конец табличного списка. Для использования инструкции нужно указать:

— название табличного списка (должно начинаться с прописной буквы латинского алфавита);

— переменные, в которых из директивы `rule` в инструкцию `insert_into` передаются значения ячеек табличного списка;

Примечание. При добавлении нескольких значений переменные имеют тип данных List. Для формирования списка значений можно использовать функцию языка XP append (см. раздел 4.7.1).

— имена колонок (с префиксом `column::`), в которые нужно добавить значения;

— значения ячеек для добавления в табличный список (тип данных должен совпадать с типом данных колонки табличного списка).

Примечание. Колонка может содержать список значений для ключевой колонки табличного списка. Если ключ из списка есть в табличном списке — значения строк перезаписываются, если нет — для них добавляются строки в конец табличного списка.

При добавлении строк в табличный список нужно указать значения ячеек для всех колонок. Если значения для колонки не указаны, результат записи зависит от свойств колонки, настроенных при создании табличного списка: если колонка может содержать null — в добавляемые ячейки будет записано значение null, если нет — появится сообщение об ошибке. Если для колонки указано одно значение, оно будет записано в ячейки всех добавляемых строк.

Формат вызова:

```
rule <Название правила>: <Условие правила>
     on <Название события> {
          append($f1, <Значение 1>)
          append($f2, <Значение 2>)
          ...
     }
     insert_into <Название табличного списка> {
          column::<Имя колонки 1> = $f1
          column::<Имя колонки 2> = $f2
          ...
     }
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие 1: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":24,
#    "status":"Open",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Нормализованное событие 2: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":25,
#    "status":"Close"
#    ...}
event Event:
     key:
          event_src.vendor
     filter {
          dst.ip == "192.0.2.11"
     }
rule Example: Event[2]
     on Event {
          $f1 = dst.ip
          $f2 = append($f2, dst.port)
          $f3 = append($f3, status)
     }
insert_into Tabular_List_CorrelationRule {
          column::IP = $f1
          column::Port = $f2
          column::Status = $f3
}
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
```

```
    $status = "success"
    $importance = "info"
}
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":24, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":25, "Status":"Close"}]
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ...}
```

10.6.5. Инструкция remove_from. Удаление строк из табличного списка

Инструкция `remove_from` предназначена для удаления строк из табличного списка. При выполнении правила корреляции удаляются строки, удовлетворяющие указанному в инструкции условию. В инструкции нужно указать название табличного списка и условие удаления строк. В условии можно использовать переменные, имена колонок табличного списка (с префиксом `column::`) и логические операторы.

Формат вызова:

```
rule <Название правила>: <Условие правила>
     ...
     remove_from <Название табличного списка> {
          <Условие для удаления>
     }
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
```

```
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
     key:
          event_src.vendor
     filter {
          event_src.vendor == "Best Company"
     }
rule Example: Event
     remove_from Tabular_List_CorrelationRule {
          column::Port > 22
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"}]
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ... }
```

10.6.6. Инструкция clear_table. Очистка табличного списка

Инструкция `clear_table` предназначена для удаления всех строк из табличного списка при выполнении правила корреляции. В инструкции нужно указать название табличного списка.

Формат вызова:

```
rule <Название правила>: <Условие правила>
     ...
     clear_table <Название табличного списка>
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11", "Port":21, "Status":"Close"},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open"},
#    {"IP":"192.0.2.11", "Port":23, "Status":"Close"}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
event Event:
     key:
          event_src.vendor
     filter {
          event_src.vendor == "Best Company"
     }
rule Example: Event
     clear_table Tabular_List_CorrelationRule
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Табличный список: "Tabular_List_CorrelationRule": []
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    ...}
```

10.7. Директива emit. Корреляционное событие

Директива `emit` предназначена для заполнения полей корреляционного события (названия полей нужно начинать со знака доллара). Директива выполняется при выполнении условия правила корреляции.

В блоке операторов директивы `emit` вы можете использовать переменные, объявленные в инструкциях `init` и `on` (названия нужно начинать со знака доллара), функции и операторы языка XP или литералы с типом данных, соответствующим заполняемым полям.

> **Внимание!** Не используйте в блоке операторов директивы `emit` поля подлежащих корреляции событий.

Формат вызова:

```
emit {
     <Блок операторов emit>
}
```

Пример

```
emit {
    $correlation_type = "incident"
    $category.generic = "Attacks & Recon"
    $category.high = "Attack"
    $category.low = "Bruteforce"
    $subject = "account"
    $action = "initiate"
    $object = "attack"
    $status = "success"
    $object.name = "Bruteforce"
    $id = "PT_SIEM_Bruteforce_success_by_user"
    $importance = "medium"
}
```

10.8. Директива query. Объявление запроса к табличному списку

Директива `query` может использоваться для объявления запроса к табличному списку. В директиве нужно указать:

- название запроса, уникальное в рамках правила корреляции. В одном правиле корреляции можно объявить несколько запросов с разными названиями к одному или нескольким табличным спискам. Обращение к запросу выполняется по его названию;
- название табличного списка, к которому выполняется запрос (должно начинаться с прописной буквы латинского алфавита);

— аргументы запроса — переменные, используемые при составлении условия запроса. Значения переменных передаются в запрос при выполнении запроса. Запрос можно выполнить с помощью функций языка XP `exec_query` или `select_query_first`;

— условие, проверяемое при выполнении запроса. В условии запроса можно использовать значения переменных, переданные в аргументах запроса, имена колонок табличного списка (с префиксом `column::`), логические операторы и функции языка XP `in_subnet`, `regex_match`.

Примечание. При использовании функции `in_subnet` в условии запроса `query` к табличному списку в первом аргументе функции нужно указать имя переменной, переданной в запрос, во втором — имя колонки табличного списка (с префиксом `column::`).

Формат вызова:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
}
```

Пример

```
# Табличный список: "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.11/24", "Port":21, "Status":"Close", "Duration":51},
#    {"IP":"192.0.2.11/24", "Port":22, "Status":"Open", "Duration":36},
#    {"IP":"192.0.2.11/24", "Port":23, "Status":"Close", "Duration":82}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.34",
#    "dst.port":22,
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Запрос TestQuery к табличному списку Tabular_List_CorrelationRule
query TestQuery ($check_ip, $check_port) from Tabular_List_CorrelationRule {
# Условие запроса возвращает True, если переданный в запрос узел находится в подсети
192.0.2.11/24 и переданный порт есть в табличном списке в состоянии Open
     in_subnet($check_ip, column::IP) and
     column::Port == $check_port and
     column::Status == "Open"
}
event Event:
# Отбираются события от источника Best Company, удовлетворяющие условию запроса
     key:
```

```
            event_src.vendor
        filter {

            event_src.vendor == "Best Company" and
            exec_query("TestQuery", [dst.ip, dst.port])
        }
rule Example: Event
        on Event {
            $time = select_query_first("TestQuery", [dst.ip, dst.port], "Duration")
            $datafield1 = "IP_" + string(dst.ip)
            $datafield2 = "Port_" + string(dst.port)
            $datafield3 = "Open_time_" + string($time)
        }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"IP_192.0.2.34",
#    "datafield2":"Port_22",
#    "datafield3":"Open_time_36",
#    ...}
```

В этом разделе

Инструкции qhandler, limit, skip (см. раздел 10.8.1)

Агрегатные функции для результатов запроса к табличному списку (см. раздел 10.8.2)

См. также

in_subnet (см. раздел 4.8.3)

Функция regex_match, проверка строки по регулярному выражению из табличного списка (см. раздел 4.9.5)

Функция exec_query, запрос к табличному списку (см. раздел 4.9.2)

Функция select_query_first, запрос значения из табличного списка (см. раздел 4.9.4)

10.8.1. Инструкции qhandler, limit, skip

Инструкция `qhandler` используется для выполнения арифметических операций над результатом запроса к табличному списку. При выполнении запроса, с инструкцией `qhandler`, функция `exec_query` возвращает ассоциативный массив (с типом данных KeyValue), содержащий результаты вычислений (с типом данных Number) для указанных в инструкции агрегатных функций. Для использования инструкции нужно указать:

— агрегатную функцию для выполнения арифметической операции;

— имя колонки табличного списка (с префиксом `column::`), к элементам которого применяется агрегатная функция (поддерживаются операции над элементами колонок с типом данных Number);

— ключ для получения результата применения агрегатной функции из ассоциативного массива, возвращаемого функцией `exec_query`.

Инструкция `limit` используется для указания количества первых строк в результате запроса, к которым применяются агрегатные функции, указанные в инструкции `qhandler`.

Инструкция `skip` используется для исключения указанного количества первых строк из результата запроса, к оставшимися строками применяются агрегатные функции, указанные в инструкции `qhandler`.

> **Примечание.** При использовании инструкций `limit` и `skip` нужно учитывать, что сортировка строк в результате запроса не определена.

Формат вызова:

```
query <Название запроса>(<Аргументы запроса>) from <Название табличного списка> {
     <Условие запроса>
     }
     qhandler {
          <Ключ 1> = <Агрегатная функция 1>(column::<Имя колонки 1>)
          <Ключ 2> = <Агрегатная функция 2>(column::<Имя колонки 2>)
          ...
     }
     limit <Максимальное число возвращаемых строк>
     skip <Количество строк, исключаемых из результата>
```

Пример

```
# Табличный список "Tabular_List_CorrelationRule": [
#    {"IP":"192.0.2.10", "Port":22, "Status":"Open", "Duration":41},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":33},
#    {"IP":"192.0.2.11", "Port":22, "Status":"Open", "Duration":67}]
# Нормализованное событие: {
#    "dst.ip":"192.0.2.11",
#    "time": "2017-10-18T12:00:00Z",
#    "id": "developer_guide",
```

```
#    "importance": "info",
#    "action": "start",
#    "object": "task",
#    "status": "success",
#    "event_src.vendor": "Best Company",
#    "event_src.title": "test",
#    "event_src.category": "Other",
#    "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
# Запрос TestQuery к табличному списку Tabular_List_CorrelationRule
query TestQuery ($check_ip) from Tabular_List_CorrelationRule {
     column::IP == $check_ip and
     column::Port == 22 and
     column::Status == "Open"
}
     qhandler {
          $get_max_duration = max(column::Duration)
          $get_min_duration = min(column::Duration)
          $get_avg_duration = avg(column::Duration)
          $get_count = count()
     }
event Event:
     key:
          event_src.vendor
     filter {
          event_src.vendor == "Best Company"
          }
rule Example: Event
     on Event {
          $get_duration = exec_query("TestQuery", [dst.ip])
          # Присваивание переменным максимального, минимального и среднего времени
открытия порта 22 для переданного в запрос IP-адреса узла
          $max_duration = $get_duration["$get_max_duration"]
          $min_duration = $get_duration["$get_min_duration"]
          $avg_duration = $get_duration["$get_avg_duration"]
          $count = $get_duration["$get_count"]
          $datafield1 = "IP_" + string(dst.ip)
          $datafield2 = "Max_duration_" + string($max_duration)
          $datafield3 = "Min_duration_" + string($min_duration)
          $datafield4 = "Average_duration_" + string($avg_duration)
          $datafield5 = "Result_count_" + string($count)
     }
emit {
    $correlation_type = "event"
    $object = "client"
    $action = "check"
    $status = "success"
```

```
    $importance = "info"
}
# Корреляционное событие: {
#    "correlation_type":"event",
#    "object":"client",
#    "action":"check",
#    "status":"success",
#    "importance":"info",
#    "datafield1":"IP_192.0.2.11",
#    "datafield2":"Max_duration_67",
#    "datafield3":"Min_duration_33",
#    "datafield4":"Average_duration_50",
#    "datafield5":"Result_count_3",
#    ...}
```

10.8.2. Агрегатные функции для результатов запроса к табличному списку

Указанные в таблице агрегатные функции используются только в инструкции `qhandler` для выполнения арифметических операций над результатами запроса к табличному списку.

Таблица 84. Агрегатные функции для выполнения арифметических операций над результатами запроса к табличному списку

Агрегатная функция	Возвращаемое значение
`min(column::<Имя колонки>)`	Наименьший элемент в результатах запроса
`max(column::<Имя колонки>)`	Наибольший элемент в результатах запроса
`avg(column::<Имя колонки>)`	Среднее значение элементов, округленное до целого
`median(column::<Имя колонки>)`	Значение медианного элемента. При четном числе элементов выбирается нижнее медианное значение
`sum(column::<Имя колонки>)`	Сумма всех элементов результатов запроса
`count()`	Количество элементов в результатах запроса
`column::<Имя колонки>`	Первый элемент из результатов запроса

11. Настройка регистрации инцидентов

Инциденты автоматически регистрируются компонентом MP 10 Core на основании одного или нескольких событий ИБ, зарегистрированных на MP SIEM Server. Инциденты регистрируются только для событий, у которых в поле `correlation_type` указано значение `incident`.

Для корреляционных событий значение поля `correlation_type` заполняется в правиле корреляции и может быть изменено с помощью правил обогащения. Также с помощью правил корреляции и обогащения можно указать значения некоторых полей регистрируемого инцидента и настроить агрегацию инцидентов.

При агрегации в один инцидент объединяются данные нескольких событий, имеющих одинаковые ключи агрегации. Ключ может быть неизменным для всех событий, регистрируемых или обогащаемых по одному правилу, или формироваться динамически для каждого события. Способ формирования ключа агрегации указывается в правиле корреляции или обогащения в поле `incident.aggregation.key`.

Для настройки регистрации инцидентов в правиле корреляции или правиле обогащения нужно:

1. Для поля `correlation_type` указать значение `incident`.
2. Настроить поля инцидента.
3. Если требуется, настроить агрегацию инцидентов.

Настройка регистрации инцидентов при регистрации корреляционных событий выполняется в правиле корреляции в директиве `emit`. При заполнении полей инцидента (названия полей нужно начинать со знака доллара) вы можете использовать переменные, объявленные в инструкциях `init` и `on` (названия нужно начинать со знака доллара), функции и операторы языка XP или литералы с типом данных, соответствующим заполняемым полям.

> **Внимание!** Не используйте в блоке операторов директивы `emit` поля подлежащих корреляции событий.

Для настройки регистрации инцидентов при обогащении корреляционных событий нужно создать правило обогащения для этих событий (или изменить уже существующие). Настройка регистрации инцидентов выполняется в директиве `enrich` в инструкции `enrich_fields`. При заполнении полей инцидента вы можете использовать переменные (названия нужно начинать со знака доллара), поля указанного в инструкции `enrich_fields` события, функции и операторы языка XP или литералы с типом данных, соответствующим заполняемым полям.

В этом разделе

См. также

Структура правила корреляции (см. раздел 10.2)

Структура правила обогащения (см. раздел 9.1)

11.1. Алгоритм регистрации инцидента

При появлении на MP SIEM Server события, в котором для поля `correlation_type` указано значение `incident`, данные этого события передаются в MP 10 Core, в том числе время регистрации события и ключ агрегации, указанные в правиле поля инцидента и параметры агрегации. В зависимости от значения ключа агрегации в MP 10 Core выполняется одно из следующих действий:

— Если ключ агрегации не указан, регистрируется новый инцидент (агрегация для этого инцидента не выполнятся).

— Если ключ агрегации указан и в базе инцидентов отсутствуют инциденты с таким же ключом, регистрируется новый инцидент с новым ключом.

— Если ключ агрегации указан и в базе инцидентов присутствуют инциденты с таким же ключом, выполняется агрегация инцидентов, данные события добавляются в инцидент. Если инцидентов с таким ключом несколько, данные события добавляются в инцидент с более поздней датой изменения.

  Примечание. Если существующий инцидент имеет статус «Закрыт», в зависимости от параметров агрегации инцидентов по событию может быть зарегистрирован новый инцидент, данные события могут быть добавлены в существующий закрытый инцидент или отброшены.

11.2. Поля для заполнения информации об инцидентах

В этом разделе приведен список полей событий для заполнения полей регистрируемых по ним инцидентов (см. документ Синтаксис языка запроса PDQL). Эти поля событий не отображаются при просмотре событий через веб-интерфейс MaxPatrol SIEM. Указанные значения полей инцидента отображаются в карточке инцидента.

incident.name

Название инцидента для заполнения поля инцидента `name`. Если значение не указано в правиле, поле инцидента будет заполнено значением из поля события `correlation_name`; если в событии значение не указано, формируется название вида `INC-<Порядковый номер>`. При агрегации инцидентов значение не меняется.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Параметры** в поле **Название**.

Тип данных: String

Чтобы заполнить имя инцидента значениями полей нормализованного события, на которое сработало правило корреляции, необходимо использовать блок операторов инструкции `on` правила корреляции для соответствующего события.

Пример:

```
on <Название события> {
    $incident.name = "Выход пользователя " + subject.name + " в выходной " + time
}
```

Если в правиле корреляции переопределяются значения как `subject.name`, так и `time`, необходимо использовать директиву `emit` с переменными корреляционного события.

Пример:

```
emit {
    $correlation_type = "incident" $incident.name = "Выход пользователя " +
$subject.name + " в выходной " + $time
}
```

incident.description

Описание инцидента для заполнения поля инцидента `description`. Если значение не указано в правиле, поле инцидента будет заполнено по правилу локализации события, если правило локализации не создано, формируется описание вида `Incident <Порядковый номер>, <Категория>`. При агрегации инцидентов значение не меняется.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Параметры** в поле **Описание**.

Тип данных: String

Пример:

```
$incident.description = "Обнаружена попытка подбора пароля для учетной записи"
```

incident.category

Тип инцидента для заполнения поля инцидента `type`. Поле нужно заполнять в соответствии с классификацией инцидентов (см. приложение Е) значениями из колонки type. По указанному значению автоматически определяется категория инцидента и заполняется поле инцидента `category` (значениями из колонки category). Если тип инцидента не указан в правиле, в полях инцидента будет `undefined`. При агрегации инцидентов значение поля не меняется.

При регистрации инцидента вручную тип инцидента выбирается в блоке параметров **Параметры** в раскрывающемся списке **Категория и тип**.

Тип данных: Enum

Пример:

```
$incident.category = "DataLeakage"
```

Примечание. Для этого примера поля инцидента примут значения `type="DataLeakage"`, `category="DataSecurity"`, в карточке инцидента будут указаны тип инцидента «Утечка данных» и категория «Безопасность данных».

incident.severity

Уровень опасности инцидента для заполнения поля инцидента `severity`. Возможные значения:

— `high` — высокая;

— `medium` — средняя (по умолчанию);

— `low` — низкая.

При регистрации инцидента вручную значение поля выбирается в блоке параметров **Статус** в раскрывающемся списке **Опасность**.

Тип данных: Enum

Пример:

```
$incident.severity = "medium"
```

incident.severity_behavior

Выбор уровня опасности инцидента при агрегации. Возможные способы заполнения поля инцидента `severity`:

— `first` — заполняется значением из поля `importance` события, по которому зарегистрирован инцидент;

— `highest` — заполняется самым высоким значением из полей `importance` всех событий, данные которых добавлены в инцидент (по умолчанию).

Тип данных: Enum

Пример:

```
$incident.severity_behavior = "first"
```

incident.assigned_to_user_id

Идентификатор пользователя MaxPatrol SIEM, который назначается ответственным за расследование инцидента. Имя и фамилия пользователя указываются в полях инцидента `assigned.FirstName` и `assigned.LastName`. При агрегации инцидентов поле заполняется значением из первого события с заполненным значением этого поля и в дальнейшем не изменяется.

При регистрации инцидента вручную значения полей выбираются в блоке параметров **Статус** в раскрывающемся списке **Ответственный**.

> **Примечание.** Для автоматического назначения ответственного вы можете создать табличный список, содержащий категории или типы инцидентов, идентификаторы и имена пользователей, и написать правило обогащения, которое будет автоматически заполнять поле события `incident.assigned_to_user_id` в зависимости от категории или типа инцидента.

Тип данных: String

Пример:

```
$incident.assigned_to_user_id = "07075c57-1b0e-481d-9017-bcb6260481f3"
```

incident.related_addresses

Названия активов, на которые направлена атака, для заполнения поля инцидента `Targets.Addresses.Name`. При агрегации инцидентов названия добавляются в список.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Активы и сети** в поле **Сетевые адреса**.

Тип данных: StringList

Пример:

```
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    on Event
        $related_addresses = src.host
emit {
    $correlation_type = "incident"
    $incident.related_addresses = append($incident.related_addresses,
$related_addresses)
}
```

incident.related_assets

Цели атаки, которые не могут быть отнесены к активам, группам активов, сетям и сетевым адресам, для заполнения поля инцидента `Targets.Others.Name`. При агрегации инцидентов адреса добавляются в список.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Активы и сети** в поле **Прочие активы и сети**.

Тип данных: StringList

Пример:

```
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    on Event
        $related_assets = src.asset
emit {
    $correlation_type = "incident"
    $incident.related_assets = append($incident.related_assets, $related_assets)
}
```

incident.attacking_addresses

Сетевые адреса — источники атаки, для заполнения поля инцидента `Attackers.Addresses.Name`. При агрегации инцидентов адреса добавляются в список.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Атакующие активы** в поле **Сетевые адреса**.

Тип данных: StringList

Пример:

```
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    on Event
        $attacking_addresses = dst.host
emit {
    $correlation_type = "incident"
    $incident.attacking_addresses = append($incident.attacking_addresses,
$attacking_addresses)
}
```

incident.attacking_assets

Источники атаки, которые не могут быть отнесены к активам, группам активов, сетям и сетевым адресам, для заполнения поля инцидента `Attackers.Others.Name`. При агрегации инцидентов адреса добавляются в список.

При регистрации инцидента вручную значение поля вводится в блоке параметров **Атакующие активы** в поле **Прочие активы и сети**.

Тип данных: StringList

Пример:

```
event Event:
    key:
        event_src.title
    filter {
        event_src.title == "test"
    }
rule Example: Event
    on Event
        $attacking_asset = dst.asset
emit {
    $correlation_type = "incident"
    $incident.attacking_assets = append($incident.attacking_assets, $attacking_asset)
}
```

11.3. Поля для настройки агрегации инцидентов

В этом разделе приведен список полей событий для настройки агрегации регистрируемых по ним инцидентов. Эти поля событий не отображаются при просмотре событий через веб-интерфейс MaxPatrol SIEM.

incident.aggregation.key

Ключ для агрегации инцидентов (регистронезависимый). При выполнении условий агрегации для событий с одинаковыми ключами регистрируется один инцидент. Если значение поля не указано в правиле, агрегация инцидентов не выполняется и для каждого события регистрируется новый инцидент. Если ранее были зарегистрированы несколько инцидентов с одинаковыми ключами, эти события добавляются в инцидент с более поздней датой изменения.

Ключ агрегации может быть неизменным для всех регистрируемых или обогащаемых по одному правилу событий или формироваться динамически для каждого события. Динамический ключ представляет собой строку, сформированную из значений указанных полей события. Вы можете использовать функцию `join` для формирования динамических ключей (см. раздел 4.5.6).

> **Внимание!** Для корректной агрегации инцидентов по событиям одного правила при формировании ключа агрегации необходимо использовать значение поля события `correlation_name`.

Тип данных: String

Пример:

```
$incident.aggregation.key = join([$correlation_name, $event_src.host], "|")
```

incident.aggregation.time_window

Выбор способа отсчета времени агрегации инцидентов, указанного в поле `incident.aggregation.timeout`. Время регистрации инцидента определяется по значению поля `time` события. Если значение поля не указано в правиле, за него принимается значение `max_aggregation_time`. Возможные значения:

— `infinite` — время агрегации не ограничено, инцидент регистрируется с ключом первого полученного события;

— `max_aggregation_time` — время агрегации отсчитывается с момента получения первого события, инцидент регистрируется при превышении времени, указанного в параметре `incident.aggregation.timeout`, и содержит все события с таким же ключом, зарегистрированные за это время (по умолчанию);

— `no_alerts_time` — время отсчитывается с момента регистрации последнего события. Инцидент регистрируется, если время между моментами регистрации событий с одинаковыми ключами превышает указанное время, — и содержит все события с момента регистрации первого события с этим ключом.

Тип данных: Enum

Пример:

```
$incident.aggregation.time_window = "infinite"
```

incident.aggregation.timeout

Время агрегации инцидентов в секундах. Для ввода времени можно использовать сокращения: `s` — секунды, `m` — минуты, `h` — часы, `d` — дни. Если значение поля не указано в правиле, за него принимается значение 2 часа.

Тип данных: Number

Пример:

```
$incident.aggregation.timeout = 1h
```

incident.aggregation.closed_behavior

Выбор действия, если зарегистрированный ранее инцидент с тем же ключом имеет статус «Закрыт». Если значение поля не указано в правиле, регистрируется новый инцидент, если при этом событие было зарегистрировано при проверке устаревших событий — данные события добавятся в зарегистрированный ранее инцидент. Возможные значения:

- `new` — регистрировать новый инцидент (по умолчанию);
- `append` — добавить новый инцидент в зарегистрированный ранее инцидент со статусом «Закрыт» (статус не изменяется). Если при этом изменяются данные инцидента, обновляется время изменения инцидента;
- `skip` — пропустить инцидент.

Тип данных: Enum

Пример:

```
$incident.aggregation.closed_behavior = "new"
```

12. Утилиты SDK

В этом разделе описаны утилиты MaxPatrol SIEM SDK, используемые для отладки правил нормализации, обогащения, корреляции и агрегации, даны примеры команд для запуска утилит, указаны необходимые для работы утилит файлы. Утилиты SDK входят в комплект поставки MaxPatrol SIEM.

Для работы с утилитами MaxPatrol SIEM SDK нужно выполнить экспорт всех объектов из Knowledge Base (для установки в SIEM Lite). При экспорте вы получите архив `knowledgebase_<Дата экспорта>.zip`, содержащий файлы правил нормализации, обогащения, корреляции и агрегации MaxPatrol SIEM, а также дополнительные файлы, необходимые для отладки новых правил.

В этом разделе

12.1. Экспорт событий из Elasticsearch. Утилита export_data

Утилита `export_data` предназначена для экспорта необработанных и нормализованных событий из Elasticsearch.

Для работы утилиты требуется, чтобы на узле MP SIEM Server была запущена служба SIEM Server frontend. Если в IT-инфраструктуре организации используются межсетевой экран или другие средств для контроля сетевого трафика, необходимо настроить в них правила, разрешающие трафик между узлом, на котором запускается утилита, и узлом MP SIEM Server через порт 8013/TCP.

Формат вызова для экспорта необработанных событий:

```
./export_data
    --time_from=<Дата и время начала интервала>
    --time_to=<Дата и время конца интервала>
    --output=<Путь к файлу событий>
    raw
    <PDQL-запрос>
```

Формат вызова для экспорта нормализованных событий:

```
./export_data
    --time_from=<Дата и время начала интервала>
```

```
--time_to=<Дата и время конца интервала>
--output=<Путь к файлу событий>
norm
<PDQL-запрос>
```

Ключи запуска утилиты:

— `--help` или `-h`

Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

— `raw`

Выполнить экспорт необработанных событий (нормализованных и ненормализованных).

Примечание. Для экспорта только ненормализованных событий вы можете запустить утилиту с ключом `raw` и PDQL-фильтром `normalized=null or normalized=false`.

— `norm`

Выполнить экспорт нормализованных событий (выполняется по умолчанию, если не указан тип событий для экспорта).

Параметры Elasticsearch:

— `--host=<IP-адрес узла>`

Узел MP SIEM Server, на котором запущена служба SIEM Server frontend.

— `--port=<Порт>`

Порт подключения к службе SIEM Server frontend (по умолчанию 8013/TCP).

Параметры запроса:

— `--time_from=<Дата и время начала интервала>` или `-f=<Дата и время начала интервала>`

Выполнить экспорт событий, зарегистрированных позднее указанных даты и времени. Дату и время нужно указывать в формате ISO 8601 (пример без поправки на часовой пояс: `2015-05-13T12:13:12Z`).

Примечание. Для ввода текущих даты и времени вы можете использовать переменную `now`, для ввода более раннего времени — конструкцию `now-<Дни>d<Часы>h`.

— `--time_to=<Дата и время конца интервала>` или `-t=<Дата и время конца интервала>`

Выполнить экспорт событий, зарегистрированных ранее указанных даты и времени. Дату и время нужно указывать в формате ISO 8601 (пример для московского времени: `2015-05-13T12:13:12+03:00`).

— `--path=<Путь к папке правил>` или `-p <Путь к папке правил>`

Выполнить экспорт событий, удовлетворяющих условиям отбора событий из правил в указанной папке. Параметр должен использоваться совместно с одним из следующих параметров:

- `--name=<Имя файла правила>` или `-n <Имя файлу правила>`

  Использовать условие отбора событий из правила из файла с указанным именем. Параметр можно указывать несколько раз с именами разных файлов.

- `--all` или `-a`

  Использовать условия отбора событий из правил из всех файлов в папке.

— `<PDQL-запрос>`

Применить к отобранным для экспорта событиям указанный PDQL-запрос (см. документ Синтаксис языка запроса PDQL). В запросе вы можете использовать поля событий, математические и логические операторы языка XP.

Примечание. Вы можете использовать в PDQL-запросе одинарные кавычки (') или экранировать двойные кавычки обратной косой чертой (\").

Выходные данные:

— `--output=<Путь к файлу событий>` или `-o <Путь к файлу событий>`

Сохранить события в указанном файле формата JSON. Каждое событие сохраняется в одной строке.

— `--zip` или `-z`

Добавить файл с событиями в архив.

— `--limit=<Значение>`

Ограничить количество событий в файле указанным значением (по умолчанию 10 000).

— `--count`

Вывести на экран количество найденных для экспорта событий.

Параметры отладки:

— `--quiet` или `-q`

Не выводить на экран информацию о работе утилиты.

— `--debug`

Вывести на экран запрос, отправляемый в Elasticsearch.

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

Пример экспорта необработанных событий

```
./export_data -f now-2d -t now --output=outfile_raw.json raw input_id =
d7b4c32b-2831-405f-9844-b6bb54ed2bf8 and tag = 'syslog' --debug 2>export_data.log
```

Пример экспорта нормализованных событий

```
./export_data -f 2015-05-13 -t 2015-05-14T12:00:00Z --output=outfile.json norm
(user.name = 'admin' and user.group = 'administrators') or id in [PT_SIEM_Cool_Id,
PT_SIEM_Other_id]
```

12.2. Отладка правил нормализации

Каждое правило нормализации сохраняется в отдельном файле `formula.xp`. Файлы правил нормализации MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `normalization-formulas\formulas`. Для отладки правил нормализации вы можете использовать утилиту `normalizer-cli`. При отладке вам могут потребоваться дополнительные файлы.

appendix.xp

Файл формата JSON, содержит правила заполнения полей нормализованного события `event_src.host`, `src.host`, `dst.host`. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip` в папке `normalization-formulas\xp_appendix`.

formulas_graph.json

Граф нормализации — файл формата JSON, описывающий совокупность всех используемых правил нормализации.

Для создания графа нормализации вы можете использовать утилиту `rcc`.

taxonomy.json

Файл формата JSON, содержит схему полей событий MaxPatrol SIEM. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip`.

В этом разделе

12.2.1. rcc

Утилита `rcc` может использоваться для создания графа нормализации.

Формат запуска:

```
rcc.exe
    --lang=normalization
    --taxonomy=<Путь к файлу taxonomy.json>
    --schema=<Путь к файлу schema.json>
    --formula-appendix=<Путь к файлу appendix.xp>
    <Путь к папке правил нормализации>
    --output=<Путь к файлу графа>
```

Ключи запуска утилиты:

— `--help` или `-h`

  Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `--lang=<Тип графа>` или `-l=<Тип графа>`

  Выбор типа правила, для которого создается граф. Для правил нормализации нужно указать `normalization` или `n`.

— `<Путь к папке правил нормализации>`

  Путь к папке с файлами правил нормализации, для которых нужно создать граф. Из папки отбираются файлы с расширением `xp`. Вместо папки вы можете указать названия файлов с правилами.

— `--taxonomy=<Путь к файлу taxonomy.json>` или `-t <Путь к файлу taxonomy.json>`

  Использовать указанный файл `taxonomy.json` со схемой полей событий.

— `--schema=<Путь к файлу schema.json>` или `-s <Путь к файлу schema.json>`

  Использовать указанный файл `schema.json` со схемой БД табличных списков.

— `--lib-path <Путь к папке макросов>` или `-p <Путь к папке макросов>`

  Путь к папке с файлами макросов. Вы можете указать несколько папок, используя параметр несколько раз в одной команде. Файлы макросов MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `filters`.

— `--formula-appendix=<Путь к файлу appendix.xp>`

Использовать указанный файл `appendix.xp` с правилами заполнения полей нормализованного события `event_src.host`, `src.host` и `dst.host`.

Выходные данные:

— `--output=<Путь к файлу графа>` или `-o <Путь к файлу графа>`

Сохранить граф нормализации в указанном файле. Если параметр не указан, граф выводится на экран (в стандартный поток вывода stdout).

— `--pretty`

Использовать для формата JSON представление с переносом строк и отступами при выводе на экран и сохранении в файл, указанный в параметре `--output`.

Параметры отладки:

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

— `--mre`

Выводить данные отладки в формате JSON. Если параметр не указан, данные выводятся в текстовом формате.

Пример

```
C:\siem-sdk\rcc.exe --lang=normalization --taxonomy=C:\siem-sdk\taxonomy.json --
schema=C:\siem-sdk\schema.json --formula-appendix=C:\siem-sdk\normalization-
formulas\xp_appendix\appendix.xp C:\knowledgebase\normalization-formulas\formulas --
output=C:\siem-sdk\formulas_graph.json
```

12.2.2. normalizer-cli

Внимание! Для работы утилиты требуются файлы библиотек `icudt55.dll`, `icuuc55.dll`.

Утилита `normalizer-cli` предназначена для нормализации потока необработанных событий. Через стандартный поток ввода данных (stdin) утилита получает необработанные события, нормализует их, используя граф нормализации, и направляет в стандартный поток вывода данных (stdout).

Формат запуска:

```
normalizer-cli.exe
    --raw
    --not-norm=<Файл для ненормализованных событий>
    --mime=<Формат события>
    <Путь к файлу formulas_graph.json>
    < <Файл с необработанными событиями>
    > <Файл для нормализованных событий>
```

Ключи запуска утилиты:

— `--help` или `-h`

Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

— `--verbose-version`

Вывести на экран (в стандартный поток вывода данных stdout) номера версий утилиты и используемых ею компонентов.

Входные данные:

— `< <Путь к файлу с событиями>`

Использовать указанный текстовый файл (стандартный поток ввода данных stdin) с необработанными событиями или, если указан параметр `--raw`, со значениями полей необработанных событий `body`. Каждое событие в файле должно быть записано в отдельной строке.

— `<Путь к файлу formulas_graph.json>`

Использовать указанный файл `formulas_graph.json` с графом нормализации.

Выходные данные:

— `> <Путь к файлу с нормализованными событиями>`

Сохранить нормализованные события (из стандартного потока вывода данных stdout) в указанном файле формата JSON.

— `--not-norm=<Путь к файлу с ненормализованными событиями>` или `-u <Путь к файлу с ненормализованными событиями>`

Сохранить в указанном файле события, которые не удалось нормализовать.

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

Параметры необработанного события:

— `--raw` или `-r`

В файле с событиями содержатся значения полей необработанных событий `body`. Формат событий нужно указать в параметре `-mime`, название модуля MP 10 Collector можно указать в параметре `--tag`:

- `--mime=<Формат>`

  Формат необработанного события:

  `text/plain` — простой текст;

  `application/json` — формат JSON;

  `application/x-pt-eventlog` — событие журнала Windows;

  `text/csv` — ассоциативный массив в формате JSON;

`text/xml` — XML.

- `--tag <tag>`

  Название модуля MP 10 Collector, получившего необработанное событие.

Параметры отладки:

— `--stat` или `-s`

Вывести на экран (в стандартный поток ошибок stderr) статистику обработанных событий, например:

```
Normalized: 151 (83.43%)
Not normalized: 30 (16.57%)
Total: 181
```

— `--trace`

Вывести на экран (в стандартный поток ошибок stderr) причину неудачной нормализации. Возможные причины: не удалось найти правило нормализации; для найденных правил нормализации не выполнено условие, объявленное после ключевого слова `COND`; возникла ошибка при распознавании текста или обработке данных события.

Примечание. Параметр можно использовать только для `--mime=text/plain`. Параметр несовместим с параметром `--benchmark`.

— `--benchmark=<Точность>` или `-b <Точность>`

Вывести в файл (в стандартный поток вывода данных stdout) отчет по измерению скорости нормализации каждого события в EPS. Рекомендуется указывать точность измерения от 100 до 10000.

Примечание. Параметр несовместим с параметром `--trace`.

Для события выводится:

```
{"eps" : <Скорость нормализации: Number>, "raw":<Текст исходного
события>,"normalized":<Текст нормализованного события>}
```

Пример

```
C:\siem-sdk\cli\normalizer-cli.exe --raw --stat --not-norm=C:\siem-sdk\not_norm.json
--mime=application/json C:\siem-sdk\formulas_graph.json < C:\siem-sdk\raw.txt > C:
\siem-sdk\norm.json
```

12.3. Отладка правил агрегации

Каждое правило агрегации сохраняется в отдельном файле `<Название правила>.agr`. Файлы правил агрегации MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `aggregation-rules`. Для отладки правил агрегации вы можете использовать утилиту `aggregator-cli`. При отладке вам могут потребоваться дополнительные файлы.

aggfilters.json

Граф агрегации — файл формата JSON, описывающий совокупность всех условий отбора событий для агрегации, указанных во всех используемых правилах агрегации.

Для создания графа агрегации вы можете использовать утилиту `rcc`.

schema.json

Файл формата JSON содержит схему для создания БД табличных списков. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip` или по аналогии создать собственный файл (см. раздел 12.6).

taxonomy.json

Файл формата JSON, содержит схему полей событий MaxPatrol SIEM. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip`.

В этом разделе

12.3.1. rcc

Утилита `rcc` может использоваться для создания графа агрегации.

Формат запуска:

```
rcc.exe
    --lang=aggregation
    --taxonomy=<Путь к файлу taxonomy.json>
    --schema=<Путь к файлу schema.json>
    <Путь к папке правил агрегации>
    --output=<Путь к файлу графа>
```

Ключи запуска утилиты:

— `--help` или `-h`

  Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `--lang=<Тип графа>` или `-l=<Тип графа>`

Выбор типа правила, для которого создается граф. Для правил агрегации нужно указать `aggregation` или `a`.

— `<Путь к папке правил агрегации>`

Путь к папке с файлами правил агрегации, для которых нужно создать граф. Из папки отбираются файлы с расширением `agr`. Вместо папки вы можете указать названия файлов с правилами.

— `--taxonomy=<Путь к файлу taxonomy.json>` или `-t <Путь к файлу taxonomy.json>`

Использовать указанный файл `taxonomy.json` со схемой полей событий.

— `--schema=<Путь к файлу schema.json>` или `-s <Путь к файлу schema.json>`

Использовать указанный файл `schema.json` со схемой БД табличных списков.

— `--lib-path <Путь к папке макросов>` или `-p <Путь к папке макросов>`

Путь к папке с файлами макросов. Вы можете указать несколько папок, используя параметр несколько раз в одной команде. Файлы макросов MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `filters`.

Выходные данные:

— `--output=<Путь к файлу графа>` или `-o <Путь к файлу графа>`

Сохранить граф агрегации в указанном файле. Если параметр не указан, граф выводится на экран (в стандартный поток вывода stdout).

— `--pretty`

Использовать для формата JSON представление с переносом строк и отступами при выводе на экран и сохранении в файл, указанный в параметре `--output`.

Параметры отладки:

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

— `--mre`

Выводить данные отладки в формате JSON. Если параметр не указан, данные выводятся в текстовом формате.

Пример

```
C:\siem-sdk\rcc.exe --lang=aggregation --taxonomy=C:\siem-sdk\taxonomy.json --
schema=C:\siem-sdk\schema.json C:\knowledgebase\aggregation-rules --output=C:\siem-
sdk\aggfilters.json
```

12.3.2. aggregator-cli

Утилита `aggregator-cli` предназначена для агрегации потока нормализованных (и корреляционных) событий, поступающих через стандартный поток ввода данных (stdin). После агрегации события направляются в стандартный поток вывода данных (stdout).

Формат запуска:

```
aggregator-cli.exe
    --rules <Путь к файлу aggfilters.json>
    --input <Путь к файлу с событиями>
    > <Путь к файлу агрегированных событий>
```

Ключи запуска утилиты:

— `--help` или `-h`

Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

— `--mode=<Режим>` или `-m <Режим>`

Выбрать режим работы утилиты. В зависимости от выбранного режима в файл с выходными данными сохраняются:

- `0` — режим Normal: агрегированные и не подвергшиеся агрегации события (выбран по умолчанию);
- `1` — режим AggregationOnly: агрегированные события;
- `2` — режим SkippedOnly: события, удовлетворяющие условию правила агрегации, но не объединенные в агрегированные события;
- `3` — режим UnfilteredOnly: события, не подвергшиеся агрегации.

Входные данные:

— `--rules=<Путь к файлу aggfilters.json>` или `-r <Путь к файлу aggfilters.json>`

Использовать граф агрегации из указанного файла `aggfilters.json`.

— `--input=<Путь к файлу с событиями>` или `-i <Путь к файлу с событиями>`

Выполнить агрегацию нормализованных (и корреляционных) событий из указанного файла (из стандартного потока ввода данных stdin) формата JSON. Одно событие может занимать только одну строку файла и должно быть ограничено фигурными скобками, разделители между событиями не используются.

Выходные данные:

— `> <Путь к файлу с выходными данными>`

Сохранять события после агрегации (из стандартного потока вывода данных stdout) в указанном файле. Если параметр не указан, события выводятся на экран.

Параметры отладки:

— `--log=<Путь к файлу журнала>` или `-l <Путь к файлу журнала>`

Сохранять журнал агрегации в указанном файле `aggregator_cli.log`.

Пример

```
C:\siem-sdk\cli\aggregator-cli.exe --rules=C:\siem-sdk\aggfilters.json --input C:
\siem-sdk\norm_events.json > C:\siem-sdk\agg_events.json --log C:\siem-
sdk\aggregator-cli.log
```

12.4. Отладка правил обогащения

Каждое правило обогащения сохраняется в отдельном файле `<Название правила>.en`. Файлы правил обогащения MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `enrichment-rules`. Для отладки правил обогащения вы можете использовать утилиту `enricher-cli`. При отладке вам могут потребоваться дополнительные файлы.

enrules_graph.json

Граф обогащения — файл формата JSON, описывающий совокупность всех условий отбора событий, указанных во всех используемых правилах обогащения.

Для создания графа обогащения вы можете использовать утилиту `rcc`.

fpta_db.db

Файл содержит БД табличных списков. Для создания и изменения БД вы можете использовать утилиту `fpta_filler`.

schema.json

Файл формата JSON содержит схему для создания БД табличных списков. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip` или по аналогии создать собственный файл (см. раздел 12.6).

taxonomy.json

Файл формата JSON, содержит схему полей событий MaxPatrol SIEM. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip`.

В этом разделе

12.4.1. fpta_filler

Утилита `fpta_filler` предназначена для создания и изменения БД табличных списков.

Формат запуска:

```
fpta_filler.exe
    --schema <Путь к файлу schema.json>
    --database <Путь к файлу БД>
    --fillType=All
```

Ключи запуска утилиты:

— `--help` или `-h`

  Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `--schema <Путь к файлу schema.json>`

  Использовать указанный файл `schema.json` со схемой БД табличных списков.

— `--database=<Путь к файлу БД>`

  Использовать указанный файл БД табличных списков `fpta_db.db`. Если файл БД отсутствует, то будет сознана новая БД в соответствии со схемой, указанной в параметре `--schema`.

— `--filesize=<Размер файла>`

  Установить максимальный размер файла БД в мегабайтах (по умолчанию 10 МБ).

Параметры фильтрации при действиях с табличными списками:

— `--table=<Название табличного списка>`

  Выполнять действие с указанным табличным списком. Параметр не используется при добавлении записей из файла, указанного в параметре `--input`.

— `--fillType=<Назначение табличного списка>`

  Примечание. Если параметр `fillType` не указан, по умолчанию выбрано значение `CorrelationRule`.

Выполнять действия с табличными списками указанного назначения: `Registry` — справочник, `CorrelationRule` — для правил корреляции, `EnrichmentRule` — для правил обогащения, `AssetGrid` — для данных об активах, `CybsiGrid` — для репутационных списков, `All` — с любыми табличными списками.

— `--filter=<PDQL-запрос>`

Выполнить действия с записями, удовлетворяющими PDQL-запросу (см. документ Синтаксис языка запроса PDQL).

Параметры записи в табличные списки:

— `--input=<Путь к файлу с записями>`

Добавить в табличные списки записи из указанного файла формата JSON (см. ниже). Если параметр не указан, табличные списки будут выведены на экран (в стандартный поток вывода данных stdout).

— `--create_tables`

Добавить новый табличный список в существующую БД.

Параметры удаления записей из табличных списков:

— `--removere`

Удалить все записи из всех табличных списков в БД. Дополнительные параметры:

- `--removere --filter=<PDQL-запрос>`

  Удалить записи, удовлетворяющие запросу, из всех табличных списков.

- `--removere --table=<Название табличного списка>`

  Удалить все записи из указанного табличного списка.

- `--removere --filter=<PDQL-запрос> --table=<Название табличного списка>`

  Удалить записи, удовлетворяющие запросу, из указанного табличного списка.

Выходные данные:

— `--output=<Путь к файлу>` или `-o <Путь к файлу>`

Сохранить табличные списки (из стандартного потока вывода данных stdout) в указанном файле формата JSON. Если параметр не указан, табличные списки будут выведены на экран.

— `--pretty`

Использовать представление с отступами при сохранении в файл формата JSON, указанный в параметре `--output`.

Параметры отладки:

— `--nofail`

Не прекращать работу утилиты при возникновении одной ошибки.

Формат файла с записями табличного списка

В файле вы можете указать записи для нескольких табличных списков. Для каждого табличного списка нужно указать название, значение ключевой колонки и значения колонок, которые не могут быть пустыми (null). Обязательные для заполнения колонки `_id` и `_last_changed` будут заполнены автоматически (в колонке `_last_changed` вы можете указать дату и время в Unix-формате).

```
{
    <Название табличного списка>:[
        {<Имя ключевой колонки>:<Значение ключа 1>, <Имя колонки 1>:<Значение 11>,
<Имя колонки 2>:<Значение 12>, ...}
        {<Имя ключевой колонки>:<Значение ключа 2>, <Имя колонки 2>:<Значение 21>,
<Имя колонки 2>:<Значение 22>, ...}
        ...
   ]
}
```

Пример:

```
{
    "Potentially_unwanted_software":[
        {"key": 6, "filename": "first_example.exe"},
        {"key": 7, "filename": "second_example.exe"}
    ],
    "LSASS_openers_whitelist": [
        {"process": "first_example.exe"},
        {"process": "second_example.exe"}
    ]
}
```

Пример просмотра записей табличного списка

Просмотр в БД `fpta_db.db` записей табличного списка Potentially_unwanted_software.

```
C:\siem-sdk\fpta_filler.exe --schema=C:\siem-sdk\schema.json --database=C:\siem-
sdk\dataSource\fpta_db.db --table=Potentially_unwanted_software
```

Пример добавления записей в табличные списки

Добавить в БД `fpta_db.db` записи из файла `input.json`.

```
C:\siem-sdk\fpta_filler.exe --schema=C:\siem-sdk\schema.json
--database=C:\siem-sdk\dataSource\fpta_db.db --input=C:\siem-sdk\input.json
```

Пример создания новой БД и добавления в нее записей

Создать БД `fpta_db.db` в соответствии со схемой `schema.json` и добавить в нее записи из файла `input.json`.

```
C:\siem-sdk\fpta_filler.exe --schema=C:\siem-sdk\schema.json --database=C:\siem-sdk\dataSource\fpta_db.db --input=C:\siem-sdk\input.json --fillType=All
```

12.4.2. rcc

Утилита `rcc` может использоваться для создания графа обогащения.

Формат запуска:

```
rcc.exe
    --lang=enrichment
    --taxonomy=<Путь к файлу taxonomy.json>
    --schema=<Путь к файлу schema.json>
    <Путь к папке правил обогащения>
    --output=<Путь к файлу графа>
```

Ключи запуска утилиты:

— `--help` или `-h`

  Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `--lang=<Тип графа>` или `-l=<Тип графа>`

  Выбор типа правила, для которого создается граф. Для правил обогащения нужно указать `enrichment` или `e`.

— `<Путь к папке правил обогащения>`

  Путь к папке с файлами правил обогащения, для которых нужно создать граф. Из папки отбираются файлы с расширением `en`. Вместо папки вы можете указать названия файлов с правилами.

— `--taxonomy=<Путь к файлу taxonomy.json>` или `-t <Путь к файлу taxonomy.json>`

  Использовать указанный файл `taxonomy.json` со схемой полей событий.

— `--schema=<Путь к файлу schema.json>` или `-s <Путь к файлу schema.json>`

  Использовать указанный файл `schema.json` со схемой БД табличных списков.

— `--lib-path <Путь к папке макросов>` или `-p <Путь к папке макросов>`

Путь к папке с файлами макросов. Вы можете указать несколько папок, используя параметр несколько раз в одной команде. Файлы макросов MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `filters`.

Выходные данные:

— `--output=<Путь к файлу графа>` или `-o <Путь к файлу графа>`

Сохранить граф обогащения в указанном файле. Если параметр не указан, граф выводится на экран (в стандартный поток вывода stdout).

— `--pretty`

Использовать для формата JSON представление с переносом строк и отступами при выводе на экран и сохранении в файл, указанный в параметре `--output`.

Параметры отладки:

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

— `--mre`

Выводить данные отладки в формате JSON. Если параметр не указан, данные выводятся в текстовом формате.

Пример

```
C:\siem-sdk\rcc.exe --lang=enrichment --taxonomy=C:\siem-sdk\taxonomy.json --
schema=C:\siem-sdk\schema.json C:\knowledgebase\enrichment-rules --output=C:\siem-
sdk\enrules_graph.json
```

12.4.3. enricher-cli

Внимание! Для работы утилиты требуются файлы библиотек `icudt55.dll`, `icuuc55.dll`.

Утилита `enricher-cli` предназначена для обогащения данными потока нормализованных и корреляционных событий.

Через стандартный поток ввода данных (stdin) утилита получает нормализованные события, обогащает их данными, используя граф обогащения, и направляет в стандартный поток вывода данных (stdout).

Формат запуска:

```
enricher-cli.exe
    -r <Путь к файлу enrules_graph.json>
    -b <Путь к БД табличных списков>
    < <Путь к файлу с событиями>
    > <Путь к файлу обогащенных событий>
```

Ключи запуска утилиты:

— `-h` или `-?`

  Вывести на экран (в стандартный поток ошибок stderr) справку утилиты.

— `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `-r <Путь к файлу enrules_graph.json>`

  Использовать указанный файл `enrules_graph.json` с графом обогащения.

— `< <Путь к файлу с событиями>`

  Выполнить обогащение нормализованных событий из указанного файла (из стандартного потока ввода данных stdin) формата JSON. Одно событие может занимать только одну строку файла и должно быть ограничено фигурными скобками, разделители между событиями не используются.

— `-b <Путь к fpta_db.db>`

  Использовать указанный файл `fpta_db.db` с БД табличных списков для правил обогащения.

Выходные данные:

— `> <Путь к файлу обогащенных событий>`

  Сохранять события после обогащения данными (из стандартного поток вывода данных stdout) в указанном файле. Если параметр не указан, события выводятся на экран.

Параметры отладки:

— `-l <Путь к файлу журнала>`

  Сохранять журнал обогащения в указанном файле `enricher-cli.log`.

— `-e <Уровень журналирования>`

  Выбрать уровень журналирования `trace`, `debug`, `info` или `fatal`.

Пример

```
C:\siem-sdk\cli\enricher-cli.exe -r C:\siem-sdk\enrules_graph.json -b C:\siem-
sdk\fpta_db.db < C:\siem-sdk\norm_events.json > C:\siem-sdk\en_events.json -e trace
-l C:\siem-sdk\enricher-cli.log
```

12.5. Отладка правил корреляции

Каждое правило корреляции сохраняется в отдельном файле `<Название правила>.co`. Файлы правил корреляции MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `correlation-rules`. Для отладки правил корреляции вы можете использовать утилиты `router-cli` и `correlator-cli`. При отладке вам могут потребоваться дополнительные файлы.

fpta_db.db

Файл содержит БД табличных списков. Для создания и изменения БД вы можете использовать утилиту `fpta_filler`.

rules_graph.json

Граф корреляции — файл формата JSON, описывающий совокупность всех последовательностей событий, приводящих к регистрации корреляционного события, указанных во всех условиях используемых правил корреляции.

Для создания графа корреляции вы можете использовать утилиту `rcc`.

schema.json

Файл формата JSON содержит схему для создания БД табличных списков. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip` или по аналогии создать собственный файл (см. раздел 12.6).

taxonomy.json

Файл формата JSON, содержит схему полей событий MaxPatrol SIEM. Вы можете использовать файл, расположенный в архиве `knowledgebase_<Дата экспорта>.zip`.

В этом разделе

12.5.1. rcc

Утилита `rcc` может использоваться для создания графа корреляции.

Формат запуска:

```
rcc.exe
    --lang=correlation
```

```
--taxonomy=<Путь к файлу taxonomy.json>
--schema=<Путь к файлу schema.json>
<Путь к папке правил корреляции>
--output=<Путь к файлу графа>
```

Ключи запуска утилиты:

— `--help` или `-h`

Вывести на экран (в стандартный поток вывода данных stdout) справку утилиты.

— `--version` или `-v`

Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `--lang=<Тип графа>` или `-l=<Тип графа>`

Выбор типа правила, для которого создается граф. Для правил корреляции нужно указать `correlation` или `c`.

— `<Путь к папке правил корреляции>`

Путь к папке с файлами правил корреляции, для которых нужно создать граф. Из папки отбираются файлы с расширением `co`. Вместо папки вы можете указать названия файлов с правилами.

— `--taxonomy=<Путь к файлу taxonomy.json>` или `-t <Путь к файлу taxonomy.json>`

Использовать указанный файл `taxonomy.json` со схемой полей событий.

— `--schema=<Путь к файлу schema.json>` или `-s <Путь к файлу schema.json>`

Использовать указанный файл `schema.json` со схемой БД табличных списков.

— `--lib-path <Путь к папке макросов>` или `-p <Путь к папке макросов>`

Путь к папке с файлами макросов. Вы можете указать несколько папок, используя параметр несколько раз в одной команде. Файлы макросов MaxPatrol SIEM вы можете найти в архиве `knowledgebase_<Дата экспорта>.zip` в папке `filters`.

Выходные данные:

— `--output=<Путь к файлу графа>` или `-o <Путь к файлу графа>`

Сохранить граф корреляции в указанном файле. Если параметр не указан, граф выводится на экран (в стандартный поток вывода stdout).

— `--pretty`

Использовать для формата JSON представление с переносом строк и отступами при выводе на экран и сохранении в файл, указанный в параметре `--output`.

Параметры отладки:

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

— `--mre`

Выводить данные отладки в формате JSON. Если параметр не указан, данные выводятся в текстовом формате.

Пример

```
C:\siem-sdk\rcc.exe --lang=correlation --taxonomy=C:\siem-sdk\taxonomy.json --
schema=C:\siem-sdk\schema.json C:\knowledgebase\correlation-rules --output=C:\siem-
sdk\rules_graph.json
```

12.5.2. router-cli

Внимание! Для работы утилиты требуются файлы библиотек `icudt55.dll`, `icuuc55.dll` и `libejdb.dll`.

Утилита `router-cli` предназначена для поиска нормализованных событий, удовлетворяющих условиям отбора событий, указанным в правилах корреляции. Через стандартный поток ввода данных (stdin) утилита получает нормализованные события, используя граф корреляции, выполняет поиск событий и направляет найденные события в стандартный поток вывода данных (stdout).

Формат запуска:

```
router-cli.exe
    -r <Путь к файлу rules_graph.json>
    < <Путь к файлу с событиями>
    > <Путь к файлу найденных событий>
```

Ключи запуска утилиты:

— `-h` или `-?`

Вывести на экран (в стандартный поток ошибок stderr) справку утилиты.

— `-v`

Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `-r <Путь к файлу rules_graph.json>`

Использовать указанный файл `rules_graph.json` с графом корреляции.

— `< <Путь к файлу с событиями>`

Выполнить поиск событий в указанном файле (в стандартном потоке ввода данных stdin) формата JSON. Одно событие может занимать только одну строку файла и должно быть ограничено фигурными скобками, разделители между событиями не используются.

— `-b <Путь к файлу fpta_db.db>`

Использовать указанный файл `fpta_db.db` с БД табличных списков для правил корреляции.

Выходные данные:

— `> <Путь к файлу найденных событий>`

Сохранят найденные события (из стандартного поток вывода данных stdout) в указанном файле. Если параметр не указан, события выводятся на экран.

— `2> <Путь к файлу ошибок>`

Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

Параметры поиска событий:

— `-k <Поле события 1>, <Поле события 2>, ...`

Искать события, относящиеся к потоку нормализованных событий, выделенному по совокупности указанных полей.

— `-f <Название правила 1>, <Название правила 2>, ...`

Искать события, удовлетворяющие условиям отбора событий в перечисленных правилах.

Параметры отладки:

— `-s`

Выводить на экран (в стандартный поток ошибок stderr) найденные события.

— `-l <Путь к файлу журнала>`

Сохранять журнал поиска в указанном файле `router-cli.log`.

— `-e <Уровень журналирования>`

Выбрать уровень журналирования `trace`, `debug`, `info` или `fatal`.

— `-t <Делитель>`

Изменить временные интервалы в условиях правил корреляции в указанное число раз. Если делитель меньше единицы (но больше нуля), интервал будет увеличен, если больше единицы — уменьшен. Например, если в условии правила корреляции с помощью оператора отсчета времени `timer` указан интервал 10 секунд, при делителе 0,5, корреляционное событие регистрируется при составлении всей последовательности событий в течение 20 секунд.

Пример

```
C:\siem-sdk\cli\router-cli.exe -r C:\siem-sdk\rules_graph.json < C:\siem-
sdk\norm_events.json > C:\siem-sdk\events.json
```

12.5.3. correlator-cli

Внимание! Для работы утилиты `correlator-cli` требуется файл библиотеки `libejdb.dll`.

Утилита `correlator-cli` предназначена для корреляции потока нормализованных событий. Через стандартный поток ввода данных (stdin) утилита получает нормализованные события, используя граф корреляции, регистрирует корреляционные события и направляет их в стандартный поток вывода данных (stdout).

Формат запуска:

```
correlator-cli.exe
    -r <Путь к файлу rules_graph.json>
    < <Путь к файлу с событиями>
    > <Путь к файлу корреляционных событий>
```

Ключи запуска утилиты:

— `-h` или `-?`

  Вывести на экран (в стандартный поток ошибок stderr) справку утилиты.

— `-v`

  Вывести на экран (в стандартный поток вывода данных stdout) номер версии утилиты.

Входные данные:

— `-r <Путь к файлу rules_graph.json>`

  Использовать указанный файл `rules_graph.json` с графом корреляции.

— `< <Путьк файлу с событиями>`

  Выполнить корреляцию нормализованных событий из указанного файла (из стандартного потока ввода данных stdin) формата JSON. Одно событие может занимать только одну строку файла и должно быть ограничено фигурными скобками, разделители между событиями не используются.

— `-b <Путь к файлу fpta_db.db>`

  Использовать указанный файл `fpta_db.db` с БД табличных списков для правил корреляции.

Выходные данные:

— `> <Путь к файлу корреляционных событий>`

  Сохранять корреляционные события (из стандартного потока вывода данных stdout) в указанном файле. Если параметр не указан, события выводятся на экран.

— `2> <Путь к файлу ошибок>`

  Сохранять данные отладки (из стандартного потока ошибок stderr) в указанном файле. Если параметр не указан, данные выводятся на экран.

Параметры корреляции:

— `-k <Поле события 1>, <Поле события 2>, ...`

  Выполнять корреляцию только для правил, содержащих в инструкции `key` директивы `event` все указанные поля события.

— `-f <Название правила 1>, <Название правила 2>, ...`

  Выполнять корреляцию только для перечисленных правил.

— `-t <Делитель>`

  Изменить временные интервалы в условиях правил корреляции в указанное число раз. Если делитель меньше единицы (но больше нуля), интервал будет увеличен, если больше единицы — уменьшен. Например, если в условии правила корреляции с помощью оператора отсчета времени `timer` указан интервал 10 секунд, при делителе 0,5, корреляционное событие регистрируется при составлении всей последовательности событий в течение 20 секунд.

Параметры отладки:

— `-s`

  Выводить на экран (в стандартный поток ошибок stderr) статистику регистрации корреляционных событий.

— `-l <Путь к файлу журнала>`

  Сохранять журнал корреляции в указанном файле `correlator_cli.log`.

— `-e <Уровень журналирования>`

  Выбрать уровень журналирования `trace`, `debug`, `info` или `fatal`.

Пример

```
C:\siem-sdk\cli\correlator-cli.exe -r C:\siem-sdk\rules_graph.json < C:\siem-
sdk\norm_events.json > C:\siem-sdk\corr_events.json -s 2> C:\siem-sdk\stderr.json -e
trace -l C:\siem-sdk\correlator_cli.log
```

12.6. Структура файла schema.json

Файл формата JSON содержит схему для создания БД табличных списков. Для каждого табличного списка нужно указать общие параметры, в зависимости от назначения табличного списка, и параметры каждой колонки. Фрагмент файла `schema.json` с параметрами табличного списка Success_bruteforce_on_host_table:

```
"Success_bruteforce_on_host_table": {
    "fields": {
        "host": {
            "type": "String",
            "primaryKey": true,
```

```
            "index": true,
            "unique": true,
            "nullable": false
        },
        "user": {
            "type": "String",
            "primaryKey": false,
            "index": false,
            "unique": false,
            "nullable": false
        },
        "_last_changed": {
            "type": "Datetime",
            "primaryKey": false,
            "index": true,
            "unique": false,
            "nullable": false
        },
        "_id": {
            "type": "Number",
            "primaryKey": false,
            "index": true,
            "unique": true,
            "nullable": false
        }
    },
    "fillType": "CorrelationRule",
    "description": "Table with list of users that logged in to the Cisco Firewall in
past 24 hours",
    "type": 1,
    "ptkbId": "942ca0ec-c076-41b0-a71e-f2eba34817e8",
    "objectId": "PT-TL-1",
    "ttl": 86400,
    "maxSize": 100000,
    "typicalSize": 80000
},
```


Параметры табличного списка типа «справочник»

Для табличного списка типа «справочник» нужно указать:

— `"fillType": "Registry"`

  Назначение табличного списка. Для табличного списка типа «справочник» параметр имеет значение `Registry`.

— `"description": "<Значение>"`

Текст с описанием табличного списка.

— `"type": <Значение>`

Тип табличного списка: `1` — стандартный, `2` — пользовательский.

— `"ptkbId": "<Системный идентификатор>"`

Системный идентификатор табличного списка.

— `"objectId": "<Идентификатор>"`

Идентификатор табличного списка.

— `"userCanEditContent": <Значение>`

Пользователь может добавлять, изменять и удалять записи в табличном списке — `true`, иначе — `false`.

Параметры табличного списка для правил корреляции и обогащения

Для табличного списка для правил корреляции и обогащения нужно указать:

— `"fillType": "<Назначение табличного списка>"`

Назначение табличного списка. Для табличного списка для правил корреляции параметр имеет значение `CorrelationRule`, для правил обогащения — `EnrichmentRule`.

— `"description": "<Значение>"`

Текст с описанием табличного списка.

— `"type": <Значение>`

Тип табличного списка: `1` — стандартный, `2` — пользовательский.

— `"ptkbId": "<Системный идентификатор>"`

Системный идентификатор табличного списка.

— `"objectId": "<Идентификатор>"`

Идентификатор табличного списка.

— `"ttl": <Значение>`

Время жизни записи табличного списка в секундах; `0` — время жизни не ограничено.

— `"maxSize": <Значение>`

Максимальное число записей.

— `"typicalSize": <Значение>`

Рекомендуемое число записей.

Параметры табличного списка для данных об активах

Для табличного списка для данных об активах нужно указать:

— `"fillType": "AssetGrid"`

Назначение табличного списка. Для табличного списка для данных об активах параметр имеет значение `AssetGrid`.

— `"description": "<Значение>"`

Текст с описанием табличного списка.

— `"type": <Значение>`

Тип табличного списка: `1` — стандартный, `2` — пользовательский.

— `"ptkbId": "<Системный идентификатор>"`

Системный идентификатор табличного списка.

— `"objectId": "<Идентификатор>"`

Идентификатор табличного списка.

— `"pdql": "<PDQL-запрос>"`

PDQL-запрос для заполнения табличного списка.

— `"pdqlMappings": {}`

Объект для указания колонок табличного списка, в которые будут записываться данные об активах, полученные из PDQL-запроса. Названия полей актива и колонок табличного списка должны совпадать с указанными в PDQL-запросе. Колонки нужно указывать через запятую в виде: `"<Название поля актива>": {"tableColumn": "<Название колонки>"}`.

Примечание. Для колонок `_id` и `_last_changed`, `complex_key` описание не требуется.

— `"groupsFilter": {"groupIds": [],"includeNested": <Значение>}`

Дополнительный фильтр по группам активов содержит объекты:

- `"groupIds": ["ID группы 1", "ID группы 2", ...]`

  Список системных идентификаторов групп активов. PDQL-запрос выполняется к данным активов в указанных группах.

- `"includeNested": <Значение>`

  Выполнять PDQL-запрос к данным активов во вложенных группах — `true`, иначе — `false` (по умолчанию).

Параметры репутационного табличного списка

Для репутационного табличного списка нужно указать:

— `"fillType": "CybsiGrid"`

Назначение табличного списка. Для репутационного табличного списка имеет значение `CybsiGrid`.

— `"description": "<Значение>"`

Текст с описанием табличного списка.

— `"type": <Значение>`

Тип табличного списка: `1` — стандартный, `2` — пользовательский.

— `"ptkbId": "<Системный идентификатор>"`

Системный идентификатор табличного списка.

— `"objectId": "<Идентификатор>"`

Идентификатор табличного списка.

— `"minScore": 90`

Параметр для ввода значения интегральной уязвимости (целое число от 0 до 100), при превышении которого объект попадет в табличный список. Если параметр не указан, в табличный список попадут все объекты.

— `"itemType": "<Значение>"`

Параметр для выбора объектов репутационного списка: `ip` — IP-адрес, `domain` — имя домена, `hash` — значение контрольной суммы;

— `"mappings": {}`

Объект для указания колонок табличного списка, в которые будут записываться объекты репутационного списка. Колонки нужно указывать через запятую в виде: `"<Название объекта репутационного списка>": {"tableColumn": "<Название колонки>"}`.

Параметры колонок

Для каждой колонки табличного списка нужно указать:

— `fields: {}`

Примечание. Каждый табличный список должен содержать колонки `_id` и `_last_changed` для идентификатора записи и времени ее изменения. Эти колонки заполняются автоматически при добавлении записи.

Объект содержит параметры колонок:

- `"<Название колонки>"`

  Название колонки.

- `"type": "<Тип данных>"`

  Тип данных колонки может принимать значения: `Datetime`, `Number`, `String`. Для хранения регулярных выражений нужно указать `Regex` (только для табличного списка типа «справочник»).

- `"primary_key": <Значение>`

  Колонка является ключевой — `true`, иначе — `false`. В табличном списке может быть одна ключевая колонка.

  Примечание. Для создания составного ключа, нужно добавить служебную колонку `complex_key` и указать в ней названия колонок, составляющих ключ. При этом для всех колонок табличного списка в параметре `primary_key` нужно указать `false`.

- `"index": <Значение>`

  Данные в колонке индексируются — `true`, иначе — `false`.

- `"unique": <Значение>`

  Значения в колонке должны быть уникальными — `true`, иначе — `false`. Для ключевой колонки должно быть `true`.

- `"nullable": <Значение>`

  Колонка может содержать null — `true`, иначе — `false`.

— Предусмотрена возможность создания служебной колонки `complex_key` для хранения параметров составного ключа. Для колонки нужно указать:

- `"type": "composite"`
- `"primary_key": true`
- `"unique": true`
- `"composite_fields":[<Название колонки 1>, <Название колонки 2>, ...]`

  Список названий колонок (не менее двух), значения которых образуют составной ключ записи.

13. Утилита PTSIEMSDK GUI

Утилита предназначена для создания и отладки отдельных правил нормализации, агрегации, обогащения, корреляции и локализации, а также для отладки их совместной работы. Добавление правил в MaxPatrol SIEM выполняется через веб-интерфейс Knowledge Base. Утилита входит в комплект поставки MaxPatrol SIEM.

Программные требования

Поддерживается работа утилиты в Windows 10, 8.1, 7 и Windows Server 2019, 2016, 2012R2, 2008R2 с установленными обновлениями.

Для работы утилиты требуется следующее ПО:

- Microsoft .NET Framework v4.7.1,
- Microsoft Visual C++ для Visual Studio 2015, 2017 и 2019 (x86),
- Microsoft Visual C++ для Visual Studio 2015, 2017 и 2019 (x64).

Примечание. Вы можете скачать это ПО с сайта support.microsoft.com.

В этом разделе

13.1. Интерфейс

Этот раздел содержит описание интерфейса утилиты PTSIEMSDK GUI.

В окне программы расположены вкладки **Редактор объектов**, **Сценарии** и **Настройка**. В нижней части окна в строке состояния указаны версии используемых утилитой MaxPatrol SIEM SDK, набора дополнительных инструментов для работы с правилами и схемы полей событий.

В этом разделе

13.1.1. Вкладка Редактор объектов

Вкладка предназначена для создания и отладки отдельных правил нормализации, агрегации, обогащения, корреляции и локализации.

В панели инструментов вкладки расположены:

— Кнопка **Создать** для создания нового правила и набора данных для его отладки.

— Кнопка **Открыть** для изменения созданного ранее правила или набора данных для его отладки.

— Кнопка **Сформировать пакет** для создания архива с правилами.

— Раскрывающийся список для быстрого выбора одного из открытых правил.

В центральной части вкладки расположены вкладки открытых правил. В названии вкладки отображается название правила, сокращение для типа правила **[N]** — для правил нормализации, **[A]** — агрегации, **[C]** — корреляции, **[E]** — обогащения, а также звездочка — если изменения в правиле не были сохранены. Вы можете менять порядок вкладок, перетаскивая их мышью в нужное положение.

> **Примечание.** В контекстном меню названия вкладки доступны пункты для копирования, переименования правила или закрытия вкладки.

В панели инструментов вкладки правила расположены:

— Меню **Файл**, содержит пункты для сохранения, копирования, переименования правила или закрытия вкладки правила.

— Меню **Правка**, содержит пункты для добавления шаблона кода правила, добавления исходных событий для тестов из файла и удаления всех тестов для правила.

— Меню **Вид**, содержит пункты для включения (отключения) переноса по словам и нумерации строк в полях вкладки.

— Кнопка  для быстрого сохранения правила.

— Кнопка **Редактор метаданных** для добавления правил локализации регистрируемых по правилу событий и описания правила.

— Кнопка **Запуск** для запуска отладки правила. По стрелке справа от кнопки вы можете выбрать тесты для отладки правила.

— Поле **Тип объекта**, в поле указан тип правила.

— Раскрывающийся список **MIME** для принудительного выбора формата исходного события (доступен только для правил нормализации).

— Кнопка  для закрытия вкладки правила.

В рабочей области вкладки правила расположены:

— Поле **Правило** (**Формула**) для ввода кода правила.

Примечание. В контекстном меню полей **Правило** (**Формула**), **Исходное событие**, **Тестовый сценарий** и **Результат** доступны пункты для работы с текстом, в том числе поиск и замена. Также в меню доступен флажок для включения переноса по словам.

— Поля **Исходное событие** или **Тестовый сценарий** для ввода исходного события для правила нормализации или тестового сценария для правил агрегации, корреляции или обогащения.

— Поле **Результат**, в котором выводятся результат и сообщения о работе правила.

— Панель для работы с тестами. В верхней части панели расположены кнопки для добавления + и для удаления − тестов. Для каждого теста нужно указать исходное событие или тестовый сценарий.

13.1.2. Вкладка Сценарии

Вкладка предназначена для отладки совместной работы правил на этапах создания графов и потоковой обработки событий при нормализации, агрегации, обогащении, корреляции и локализации.

В панели инструментов вкладки расположены:

— Кнопка **Запуск** для запуска сценария.

— Кнопка **Просмотр событий** для просмотра локализованных событий из указанного файла.

— Кнопка **Папка с результатами** для просмотра папки с результатами выполнения сценария в проводнике Windows. По кнопке открывается папка, указанная на вкладке **Настройка** → **Пути** в блоке параметров **Прочее** в поле **Папка с результатами**.

— Кнопка **Сброс параметров** для очистки блоков параметров, соответствующих этапам совместной работы правил.

В рабочей области вкладки расположена панель **Параметры BUILD**, которая содержит флажки с названиями этапов, связанных с созданием графов нормализации, агрегации, обогащения, корреляции и локализации, а также с созданием базы данных табличных списков. Установка того или иного флажка включает соответствующий этап в сценарий и делает доступным блок для настройки параметров выполнения этого этапа. В панели доступны следующие флажки:

— **BUILD FORMULAS** — для настройки создания графа нормализации. В блоке нужно указать расположение папки с правилами нормализации, для которых нужно создать граф. Из указанной и всех вложенных папок отбираются файлы с расширением `xp`. Граф нормализации сохраняется в папке с результатами в файле `formulas_graph.json`.

— **BUILD AGGREGATION RULES** — для настройки создания графа агрегации. В блоке нужно указать расположение папки с правилами агрегации, для которых нужно создать граф. Из указанной и всех вложенных папок отбираются файлы с расширением `agr`. Граф агрегации сохраняется в папке с результатами в файле `aggfilters.json`.

— **BUILD TABLE LISTS DATABASE** — для настройки создания БД табличных списков. В блоке нужно указать расположение файла `schema.json` со схемой табличных списков и файла `correlation_defaults.json` с записями для заполнения табличных списков. БД сохраняется в папке с результатами в файле `fpta_db.db`.

— **BUILD ENRICHMENT RULES** — для настройки создания графа обогащения. В блоке нужно указать расположение папки с правилами обогащения, для которых нужно создать граф. Из указанной и всех вложенных папок отбираются файлы с расширением `en`. Также нужно указать расположение файла `schema.json` со схемой табличных списков. Граф обогащения сохраняется в папке с результатами в файле `enrules_graph.json`.

— **BUILD CORRELATION RULES** — для настройки создания графа корреляции. В блоке нужно указать расположение папки с правилами корреляции, для которых нужно создать граф. Из указанной и всех вложенных папок отбираются файлы с расширением `co`. Также нужно указать расположение файла `schema.json` со схемой табличных списков. Граф корреляции сохраняется в папке с результатами в файле `corrules_graph.json`.

— **BUILD LOCALIZATION** — для настройки создания графов локализации нормализованных, агрегированных и корреляционных событий для русского и английского языков. В блоке нужно указать расположение папок с правилами локализации правил в форматах YAML или XML. Из указанных и всех вложенных папок отбираются файлы `i18n_ru.yaml`, `i18n_en.yaml` или `i18n_ru.xml`, `i18n_en.xml`. Графы локализации сохраняются в папке с результатами в папке `langs` в файлах `ru.lang` и `en.lang`.

В рабочей области вкладки расположена панель **Параметры RUN**, которая содержит флажки с названиями этапов обработки событий: нормализации, агрегации, обогащения, корреляции и локализации. Установка флажка включает соответствующий этап в сценарий и делает доступными блок для настройки параметров выполнения этого этапа. В панели доступны следующие флажки:

— **NORMALIZE** — для настройки нормализации событий. В блоке нужно указать расположение файлов с графом нормализации `formulas_graph.json` и с исходными событиями. Для сохранения ненормализованных событий в файле `not_normalized.json` нужно установить флажок **Создать файл ненормализованных событий**, для вывода статистики нормализации — установить флажок **Показать статистику**. Нормализованные события сохраняются в папке с результатами в файле `norm_events.json`.

  Примечание. В файле с исходными событиями можно указывать либо события в формате JSON (прошедшие предварительную обработку в MaxPatrol SIEM), либо необработанные события. Каждое событие нужно указывать в отдельной строке. Если в файле указаны необработанные события, нужно установить флажок **Исходные события без конвертов** и с помощью раскрывающегося списка **MIME** выбрать тип событий. Все необработанные события в файле должны быть одного типа.

— **AGGREGATE** — для настройки агрегации событий. В блоке нужно указать расположение файлов с графом агрегации `aggfilters.json` и с нормализованными событиями. Агрегированные события сохраняются в папке с результатами в файле `aggr_norm_events.json`.

— **ENRICH** — для настройки обогащения событий. В блоке нужно указать расположение файлов с графом обогащения `enrules_graph.json` и с нормализованными событиями. Если в правилах обогащения используются запросы к табличным спискам, нужно указать расположения файла `fpta_db.db` с БД табличных списков. После обогащения события сохраняются в папке с результатами в файле `enrich_events.json`.

— **CORRELATE** — для настройки корреляции событий. В блоке нужно указать расположение файлов с графом корреляции `corrules_graph.json` и с нормализованными событиями. Если в правилах корреляции используются запросы к табличным спискам, нужно указать расположения файла `fpta_db.db` с БД табличных списков. Корреляционные события сохраняются в папке с результатами в файле `corr_events.json`. Для вывода в окне **Сценарии** статистики корреляции нужно установить флажок **Показать статистику**.

Примечание. Вы можете настроить фильтр правил, по которым будет выполнятся корреляция, установив флажок **Дополнительные фильтры** и указав через запятую названия правил или полей событий. Для корреляции по правилу все указанные поля событий должны содержаться в правиле корреляции в инструкции `key` директивы `event`.

— **LOCALIZE** — для настройки локализации событий. В блоке нужно указать расположение папки с файлами графов локализации `ru.lang`, `en.lang` и файла с нормализованными событиями. Описания регистрируемых по правилам событий сохраняются в папке с результатами в файлах `ru_events.json` и `en_events.json`. Чтобы после завершения сценария просмотреть описания зарегистрированных событий, нужно установить флажок **Автоматически запускать просмотр событий**.

13.1.3. Вкладка Настройка

Вкладка предназначена для настройки параметров работы утилиты и содержит вкладки **Пути**, **Редактор объектов**, **Сценарии** и **Общие**. В верхней части вкладки расположена кнопка **О приложении** для просмотра информации об утилите.

Вкладка Пути

Вкладка **Пути** содержит следующие блоки параметров:

— **Исходный код объектов** — для выбора папок с правилами для отладки их совместной работы:

- **Папки с объектами (по умолчанию)** — поле для выбора папок с правилами.
- **Папка с макросами** — поле для выбора папки с файлами фильтров для правил обогащения и корреляции.
- **Файл «appendix.xp»** — поле для выбора файла `appendix.xp`, содержащего правила заполнения полей нормализованного события `event_src.host`, `src.host`, `dst.host`.

— **Инструменты и файлы** — для выбора инструментов и файлов, необходимых для работы утилиты:

- **Папка «ptsiem-sdk»** — поле для выбора папки с утилитами MaxPatrol SIEM SDK (входит в комплект поставки утилиты).
- **Папка «build-tools»** — поле для выбора папки с набором дополнительных инструментов для работы с правилами (входит в комплект поставки утилиты).
- **Файл «taxonomy»** — поле для выбора файла `taxonomy.json` со схемой полей событий MaxPatrol SIEM (входит в комплект поставки утилиты).
- **Файл схемы табличных списков** — поле для выбора файла `schema.json` со схемой табличных списков.
- **Файл значений по умолчанию для табличных списков** — поле для выбора файла `correlation_defaults.json` с записями для заполнения табличных списков.

— **Прочее** — содержит поле **Папка с результатами** для выбора папки сохранения результатов совместной работы правил.

Вкладка Редактор объектов

Вкладка содержит следующие блоки параметров для настройки редактора объектов:

— **Цветовая тема** — для выбора светлого или темного фона полей.

— **Шрифт** — для изменения шрифта текста в полях.

— **Дополнительно** — для настройки дополнительных параметров отображения кода. В блоке расположены следующие флажки:

- **Подсветка синтаксиса** — при установленном флажке операторы, функции, токены, ключевые слова, поля и значения полей событий выделяются в коде цветом.
- **Подсветка совпадающих слов** — при установленном флажке во время ввода кода одинаковые слова выделяются серым.

- **Сворачивание кода** — при установленном флажке появляется возможность сворачивать блоки кода инструкций `filter`, `init`, `on`, `emit`, функции `submessage` и конструкции `subformula ... endsubformula`.
- **Автоотступ** — при установленном флажке по клавише Enter выполняется выравнивание строк кода.
- **Автозаполнение** — при установленном флажке **Фрагменты кода** во время ввода кода выводятся подсказки для названий функций, при установленном флажке **Описание функций** — краткие описания функций.

— **Образец** — для просмотра примера отображения кода правил.

— **Сеанс пользователя** — для настройки автоматического сохранения правил и восстановления результата предыдущего сеанса работы.

— **Параметры по умолчанию** — для настройки параметров создания и сохранения правил:

- **Скрыть вкладки** — при установленном флажке не отображаются вкладки открытых правил. Выбор правила осуществляется с помощью раскрывающегося списка в панели инструментов.
- **Добавлять шаблон метаданных для новых объектов** — при установленном флажке при создании набора данных для отладки правила в набор автоматически добавляется шаблон для метаданных.
- **Проверять состояние тестов перед сохранением объектов** — при установленном флажке при сохранении тестов для отладки правил выполняется проверка корректности выполнения тестов. Если тесты не выполнялись или завершились с ошибкой, выводится предупреждение.

— **Вывод результата** — для настройки сортировки полей событий в поле **Результат**.

— **Отображение названия объектов** — для настройки названий вкладок:

- **Отображать тип объекта в названии** — при установленном флажке в названиях вкладок отображаются сокращения для типов правил.
- **Помечать звездочкой (*) несохраненные объекты** — при установленном флажке для несохраненного правила в названии вкладки отображается звездочка (*).

Вкладка Сценарии

Вкладка **Сценарии** содержит блок параметров **Сеанс пользователя** с флажками для настройки автоматического сохранения сценария и восстановления результата предыдущего сеанса работы.

Вкладка Общие

Вкладка содержит блок параметров **Язык** с раскрывающимся списком для выбора языка интерфейса утилиты.

13.2. Предварительная настройка

Перед началом работы с утилитой рекомендуется выполнить экспорт всех объектов из Knowledge Base. При экспорте вы получите архив, содержащий файлы правил нормализации, обогащения, агрегации, корреляции и локализации, используемых в MaxPatrol SIEM, а также дополнительные файлы, необходимые для отладки новых правил. После экспорта нужно распаковать архив в папку `knowledgebase_<Дата экспорта>`.

- Чтобы настроить утилиту:

  1. Запустите файл `PTSIEMSDK_GUI.exe`.

     Откроется окно утилиты PTSIEMSDK GUI.

  2. Выберите вкладку **Настройка**.

  3. Выберите вкладку **Пути**.

  4. В блоке параметров **Прочее** справа от поля **Папка с результатами** нажмите … и выберите папку для сохранения результатов работы утилиты.

  5. Если вы планируете выполнять отладку совместной работы правил, в поле **Папки с объектами** по кнопке **Добавить папку** добавьте папки с вашими правилами.

  6. Если вы планируете выполнять отладку совместной работы правил с правилами MaxPatrol SIEM, по кнопке **Добавить папку** добавьте папки `knowledgebase_<Дата экспорта>\aggregation-rules`, `knowledgebase_<Дата экспорта>\correlation-rules\rules\correlation`, `knowledgebase_<Дата экспорта>\enrichment-rules`, `knowledgebase_<Дата экспорта>normalization-formulas\formulas`.

  7. Если вы планируете выполнять отладку правил нормализации, в блоке параметров **Исходный код объектов** справа от поля **Файл «appendix.xp»** нажмите … и выберите файла `appendix.xp`.

     Вы можете выбрать файл, расположенный в папке `knowledgebase_<Дата экспорта>\normalization-formulas\xp_appendix`.

  8. Если вы планируете выполнять отладку правил корреляции или обогащения и в них используются макросы, справа от поля **Папка с макросами** нажмите … и выберите папку с файлами макросов.

     Если вы используете только макросы из MaxPatrol SIEM, вы можете выбрать папку `knowledgebase_<Дата экспорта>\filters`.

  9. Если вы планируете выполнять отладку правил корреляции или обогащения и в них используются запросы к табличным спискам, в блоке параметров **Инструменты и файлы** справа от поля **Файл схемы табличных списков** нажмите … и выберите файл `schema.json`, справа от поля **Файл значений по умолчанию для табличных списков** нажмите … и выберите файл `correlation_defaults.json`.

Если вы используете только табличные списки из MaxPatrol SIEM, вы можете выбрать файлы, расположенные в папке `knowledgebase_<Дата экспорта>`.

Утилита настроена.

13.3. Отладка отдельного правила

Внимание! После отладки отдельного правила нормализации необходимо убедиться в отсутствии конфликтов с другими правилами MaxPatrol SIEM, после отладки отдельного правила агрегации, обогащения или корреляции — проверить работу правила с потоком событий. Для этого нужно выполнить отладку совместной работы правил.

Вы можете выполнять отладку отдельных правил нормализации, агрегации, обогащения, корреляции и локализации. После отладки правила вы можете сохранить набор данных для отладки в отдельной папке с названием правила. При этом сохраняются файлы с кодом правила, с исходными событиями, с описанием правила и с правилами локализации для различных языков. Для отладки отдельного правила нужно:

1. Создать набор данных для отладки правила и добавить в него код правила.
2. Добавить в набор исходное событие (для правила нормализации) или тестовый сценарий (для правил обогащения, корреляции или агрегации).
3. Если необходимо выполнить отладку правила для нескольких типов необработанных событий или нескольких сценариев — добавить в набор дополнительные тесты для каждого типа или сценария.
4. Если вы планируете выполнять отладку совместной работы правил локализации — добавить в набор метаданные для локализации.
5. Запустить отладку правила.

В этом разделе

Создание набора данных для отладки и отладка правила (см. раздел 13.3.1)

Составление тестового сценария (см. раздел 13.3.2)

Отладка правила с помощью тестов (см. раздел 13.3.3)

Метаданные для локализации (см. раздел 13.3.4)

13.3.1. Создание набора данных для отладки и отладка правила

- Чтобы создать набор данных для отладки и выполнить отладку правила:

  1. Запустите файл `PTSIEMSDK_GUI.exe`.

Откроется окно утилиты PTSIEMSDK GUI.

2. Выберите вкладку **Редактор объектов**.
3. В панели инструментов нажмите кнопку **Создать**.

   Откроется окно **Создание объекта**.
4. В поле **Название** введите название набора данных для отладки правила (совпадает с названием правила).
5. В раскрывающемся списке **Тип** выберите тип правила.
6. Нажмите кнопку **Создать**.

   Появится вкладка **<Название правила>**.
7. В поле **Формула** или **Правило** введите код правила.

   Примечание. Для ускорения ввода кода правила вы можете использовать шаблон, выбрав в меню **Правка** пункт **Вставить шаблон** → **<Название шаблона>**.
8. Введите данные для отладки правила:
   - Если нужно выполнить отладку правила нормализации, в поле **Исходное событие** введите событие.
   - Если нужно выполнить отладку правила агрегации, обогащения или корреляции, в поле **Тестовый сценарий** введите тестовый сценарий.
9. Нажмите кнопку **Запуск**.

   При корректной работе правила в поле **Результат** появятся поля зарегистрированного события, при некорректной — описание ошибки.
10. В панели инструментов вкладки нажмите  .

    Откроется окно **Сохранить как**.
11. В поле **Папка** укажите папку для сохранения набора данных для отладки правила.
12. В поле **Название** введите название правила.
13. Нажмите кнопку **Сохранить**.

Набор данных для отладки создан.

13.3.2. Составление тестового сценария

Тестовые сценарии используются для отладки правил агрегации, обогащения и корреляции. Тестовый сценарий должен содержать хотя бы одну строку с нормализованным событием, хотя бы одну строку с критерием корректной работы правила и может содержать строку с записями для наполнения табличных списков, используемых в правиле.

Нормализованные события

Для отладки правила нужно указать нормализованные события — такие, чтобы их особенности, количество и порядок приводили хотя бы к одному срабатыванию правила. Каждое событие нужно вводить в отдельной строке в фигурных скобках (в формате JSON). Для каждого события нужно указать значения обязательных полей. Пример:

```
{"dst.port":22, "time": "2017-10-18T12:00:00Z", "id": "developer_guide",
"importance": "info", "action": "start", "object": "task", "status": "success",
"event_src.vendor": "Best Company", "event_src.title": "test", "event_src.category":
"Other", "uuid": "00000005-9f0a-0e90-f000-0000abbac077"}
```

Ожидаемый результат, ключевое слово expect

Критерием корректной работы правила является количество событий, которые должны быть зарегистрированы на основе переданных в правило нормализованных событий. Для ввода критерия используется ключевое слово `expect`. Вы можете указать целое положительное число событий, `any` — если количество событий должно быть не менее одного и `not` — если ни одного события не должно быть зарегистрировано. Вы также можете указать шаблон регистрируемых по правилу событий (в формате JSON). В фигурных скобках нужно через запятую перечислить названия и значения полей, которые должны содержаться в событиях. Если вы хотите указать шаблоны для нескольких событий, вы можете добавить несколько строк с ключевым словом `expect`.

Формат ввода критерия:

```
expect <Количество событий 1> {<Шаблон события 1>}
expect <Количество событий 2> {<Шаблон события 2>}
...
```

Например, если по правилу агрегации `Connection_teardown_UDP` должно быть зарегистрировано одно событие, для проверки вы можете указать:

```
expect 1 {"aggregation_name": "Connection_teardown_UDP"}
```

Ожидаемое изменение табличных списков, ключевые слова expect table_list

> **Примечание.** Все используемые в правиле табличные списки должны быть описаны в файле `schema.json`.

Если при срабатывании правила выполняется запись в табличный список, то критерием корректной работы правила также является изменение табличного списка. Для ввода ожидаемых результатов изменения табличного списка используются ключевые слова `expect table_list`. В отдельной строке в фигурных скобках в формате JSON для каждого табличного списка нужно указать записи, которые должны появиться в списке. Нужно указать значения для всех колонок табличного списка.

Формат ввода записей табличного списка:

```
"<Название табличного списка>":[{"<Название колонки 1>":"<Значение колонки 11>",
"<Название колонки 2>":"<Значение колонки 21>, ...},{"<Название колонки
1>":"<Значение колонки 12>", "<Название колонки 2>":"<Значение колонки
22>, ...}, ...]
```

Формат ввода критерия:

```
expect table_list {"<Название табличного списка 1>":[<Записи табличного списка 1>],
{"<Название табличного списка 2>":[<Записи табличного списка 2>], ...}
```

Например, если после выполнения правила в табличном списке Incidents должна появиться запись с данными об узле и имени пользователя, для проверки вы можете указать:

```
expect table_list {"Incidents":[{"host": "win10x64-130.example.com", "user_name":
"test-admin"}]}
```

Наполнение табличных списков, ключевое слово table_list

Если в правиле используются табличные списки, вы можете указать записи, которыми нужно наполнить эти табличные списки для отладки правила. Для добавления записей используется ключевое слово `table_list`. Если в правиле нужно использовать записи, добавляемые в списки по умолчанию (из файла `correlation_defaults.json`), вы можете указать `table_list default`. Если для некоторых табличных списков нужно использовать другие записи, укажите эти записи для каждого табличного списка в формате, который описан выше.

Формат ввода критерия:

```
table_list default
table_list {"<Название табличного списка 1>":[<Записи табличного списка 1>],
{"<Название табличного списка 2>":[<Записи табличного списка 2>], ...}
```

Например, для наполнения табличных списков MITRE_ATTCK_whitelist и AD_Security_Administrators вы можете указать:

```
table_list {"MITRE_ATTCK_whitelist":[], "AD_Domain_Controllers":[{"hostname":
"win10x64-010.example.local", "ip": "192.0.2.10"}], "AD_Security_Administrators":
[{"username": "test-admin", "source_ip": "192.0.2.11"}]}
```

См. также

Структура файла schema.json (см. раздел 12.6)

13.3.3. Отладка правила с помощью тестов

- Чтобы выполнить отладку правила с помощью тестов:

  1. Запустите файл `PTSIEMSDK_GUI.exe`.

     Откроется окно утилиты PTSIEMSDK GUI.

  2. Выберите вкладку **Редактор объектов**.

  3. В панели инструментов нажмите кнопку **Открыть**.

Откроется окно **Открытие объектов**.

4. В поле **Папка с объектами** введите путь к папке, которая содержит вложенные папки с правилами.
5. В блоке параметров **Обзор объектов** в раскрывающемся списке выберите тип правила.

   Появится список папок с правилами.

   Примечание. Вы можете использовать быстрый поиск по названию папки правила.
6. Выберите папку и нажмите кнопку **Открыть**.

   Появится вкладка **<Название правила>** с кодом правила. Если в выбранной папке содержатся данные для отладки правила, они также появятся на вкладке.
7. В панели тестов нажмите + .

   Примечание. Для правила нормализации вы можете добавить одновременно несколько тестов с исходными событиями из файла, используя пункт меню **Правка** → **Импортировать исходные события** → **Из одного файла**.

   В панели появится строка со значком ⊕ и номером теста.
8. В поле **Исходное событие** или **Тестовый сценарий** введите необработанное событие или тестовый сценарий.

   Примечание. Вы можете удалить тест выделив строку с его номером и нажав — .
9. Если нужно, аналогичным образом добавьте другие тесты.
10. Нажмите кнопку **Запуск**.

    Последовательно будут запущены все добавленные тесты. Для просмотра результата в панели тестов выберите строку теста. Если тест пройден успешно, в строке теста появится значок ✓, в поле **Результат** отобразятся поля зарегистрированного события; если нет — появится значок ✗, в поле **Результат** отобразится описание ошибки.
11. В панели инструментов вкладки нажмите 💾.

13.3.4. Метаданные для локализации

Вы можете добавлять метаданные для локализации в наборы данных для отладки правил нормализации, обогащения, агрегации и корреляции. Метаданные используются для хранения правил локализации регистрируемых по правилам событий и описаний правил. На основе правил локализации создается граф локализации, по которому в соответствии с указанным в правиле условием для каждого зарегистрированного в MaxPatrol SIEM события в интерфейсе отображается его описание.

Вы можете вводить метаданные для правила в окне **Редактор метаданных**, используя разметку YAML. В папке каждого правила для хранения метаданных создаются: файл `metainfo.yaml` для хранения правил локализации и папка `i18n` с файлами `i18n_ru.yaml` и `i18n_en.yaml` для хранения описания правила и описаний событий.

Поле Метаданные

Поле **Метаданные** используется для ввода условий правил локализации в следующем виде:

```
EventDescriptions:
    - Criteria: <Условие правила локализации 1>
      LocalizationId: <Ключ правила локализации 1>
    - Criteria: <Условие правила локализации 2>
      LocalizationId: <Ключ правила локализации 2>
...
```

Секция `Criteria` предназначена для ввода условия правила локализации. Условие может состоять из одного или нескольких предикатов вида: `<Поле события><Оператор><Значение>` или `<Поле 1 события><Оператор><Поле 2 события>`. В предикате вы можете использовать логические операторы равенства (`=`) и неравенства (`!=`). Между предикатами можно использовать операторы `and` или `or`. В условии можно использовать круглые скобки.

> **Примечание.** В условии вы также можете использовать предикаты вида: `<Поле события>` и `not <Поле события>` (альтернативная запись `!<Поле события>`). Значение первого предиката будет `true`, если значение поля события отличается от `null`, второго — если значение равно `null` или `false`.

Секция `LocalizationId` предназначена для ввода идентификатора (ключа) для сопоставления условия правила локализации с одним описанием событий, указанным в поле **RU**, и одним описанием, указанным в поле **EN**.

Пример:

```
EventDescriptions:
    - Criteria: correlation_name = "Bruteforce_attempt_from_src" and correlation_type
= "incident"
      LocalizationId: Bruteforce_attempt_from_src_incident
    - Criteria: correlation_name = "Bruteforce_attempt_from_src" and correlation_type
= "event"
      LocalizationId: Bruteforce_attempt_from_src_incident_event
```

Поля RU и EN

Поля **RU** и **EN** используются для ввода описания правила и описаний регистрируемых по правилу событий в следующем виде:

```
Description: <Описание правила>
EventDescriptions:
    - LocalizationId: <Ключ правила локализации 1>
      EventDescription: <Описание события 1>
    - LocalizationId: <Ключ правила локализации 2>
```

```
      EventDescription: <Описание события 2>
...
```

Секция `Description` предназначена для ввода описания правила.

> **Примечание.** При использовании в описании двоеточия все описание нужно заключать в кавычки.

Секция `EventDescriptions` предназначена для ввода описаний регистрируемых по правилу событий для разных условий локализации.

Секция `EventDescription` предназначена для ввода описания регистрируемых по правилу событий.

Секция `LocalizationId` предназначена для ввода идентификатора (ключа) для сопоставления описания событий с условием правила локализации, указанным в поле **Метаданные**.

Пример для поля **RU**:

```
Description: Обнаружение большого количества ошибок при попытках войти в систему с одного узла и попытке подбора пароля
EventDescriptions:
    - LocalizationId: Bruteforce_attempt_from_src_incident
      EventDescription: Обнаружена попытка подбора пароля с узла {src.host}
    - LocalizationId: Bruteforce_attempt_from_src_incident_event
      EventDescription: Обнаружено большое количество ошибок при попытках войти в систему с узла {src.host}
```

Пример для поля **EN**:

```
Description: Large number of errors during attempt to log in from host and password bruteforce detection
EventDescriptions:
    - LocalizationId: Bruteforce_attempt_from_src_incident
      EventDescription: An attempt of host {src.host} to bruteforce a password is detected
    - LocalizationId: Bruteforce_attempt_from_src_incident_event
      EventDescription: Large number of errors is detected during attempts to log in to the system from host {src.host}
```

Добавление метаданных

- Чтобы добавить метаданные в набор данных для отладки правила:

1. Запустите файл `PTSIEMSDK_GUI.exe`.

   Откроется окно утилиты PTSIEMSDK GUI.

2. Выберите вкладку **Редактор объектов**.
3. В панели инструментов нажмите кнопку **Открыть**.

   Откроется окно **Открытие объектов**.

4. В поле **Папка с объектами** укажите путь к папке с правилом.

Появится вкладка **<Название правила>** с кодом правила. Если в выбранной папке содержатся данные для отладки правила, они также появятся на вкладке.

5. В панели инструментов вкладки нажмите кнопку **Редактор метаданных**.

   Откроется окно **Редактор метаданных**.

6. В нижней правой части окна в меню **Параметры** выберите пункт **Вставить шаблон**.
7. В поле **Метаданные** введите условие правила локализации событий.
8. В поле **RU** введите описание правила и описание событий на русском языке.
9. В поле **EN** введите описание правила и описание событий на английском языке.
10. В панели инструментов вкладки нажмите  .

Метаданные добавлены в наборы данных для отладки правила.

13.4. Отладка совместной работы правил

После создания и отладки отдельных правил перед добавлением их в MaxPatrol SIEM необходимо выполнить отладку совместной работы правил. Это позволит проверить работу правила с потоком событий и убедиться в отсутствии конфликтов с другими правилами. Вы можете гибко настроить сценарий отладки, выбрав нужные этапы, или проверить совместную работу всех доступных правил на всех этапах обработки событий. Файлы с результатами совместной работы правил сохраняются в указанной вами папке для дальнейшего анализа.

Если вы создали правила нормализации, необходимо выполнить отладку совместной работы ваших правил с правилами MaxPatrol SIEM, чтобы убедиться в отсутствии конфликтов. Для этого нужно настроить сценарий для создания графа нормализации и нормализации событий по вашим правилам и правилам MaxPatrol SIEM. Если результат нормализации совпадает с тем, который вы получили для тех же исходных событий при отладке отдельного правила, — конфликта нет.

Если вы создали правила для других этапов обработки событий, необходимо для этих этапов выполнить отладку совместной работы ваших правил с правилами MaxPatrol SIEM, чтобы проверить работу правил с потоком событий. Например, если вы создали правила обогащения и корреляции, нужно настроить сценарий для создания базы данных табличных списков, графов обогащения и корреляции, обогащения и корреляции событий по вашим правилам и правилам MaxPatrol SIEM. Если результат корреляции совпадает с тем, который вы получили для тех же исходных событий при отладке отдельных правил, — правила корректно работают с потоком событий.

В этом разделе

Сценарий для отладки правил нормализации и локализации (см. раздел 13.4.1)

Сценарий для отладки правил для всех этапов обработки событий (см. раздел 13.4.2)

13.4.1. Сценарий для отладки правил нормализации и локализации

- Чтобы настроить сценарий для отладки совместной работы правил нормализации и локализации:

  1. Запустите файл `PTSIEMSDK_GUI.exe`.

     Откроется окно утилиты PTSIEMSDK GUI.

  2. Выберите вкладку **Сценарии**.
  3. В панели **Параметры BUILD** установите флажок **BUILD FORMULAS**.
  4. В панели **Параметры RUN** установите флажок **NORMALIZE**.
  5. Справа от поля **Файл исходных событий** нажмите ... и выберите файл с исходными событиями.
  6. Если исходные события не были предварительно обработаны в MaxPatrol SIEM и преобразованы в формат JSON, установите флажок **Исходные события без конвертов** и в раскрывающемся списке **MIME** выберите тип событий.
  7. Установите флажок **Показать статистику**.
  8. Установите флажок **Создать файл ненормализованных событий**.
  9. В панели **Параметры BUILD** установите флажок **BUILD LOCALIZATION**.
  10. В панели **Параметры RUN** установите флажок **LOCALIZE**.
  11. Установите флажок **Автоматически запускать просмотр событий**.
  12. В панели инструментов нажмите кнопку **Запуск**.

      Откроется окно **Сценарии**, начнутся создание графов нормализации и локализации, нормализация и локализация исходных событий. По окончании нормализации в окне появится статистика по событиям.

  13. В окне **Сценарии** нажмите кнопку **Продолжить**.

      Откроется окно **Просмотр событий** с нормализованными событиями.

  14. Закройте окно.
  15. В панели инструментов нажмите **Папка с результатами**.

      В открывшейся папке содержатся созданные файлы: `formulas_graph.json` — граф нормализации; `ru_events.json` и `en_events.json` — графы локализации; `not_normalized.json` — события, которые не удалось нормализовать; `norm_events.json` — нормализованные события; `ru_events.json` и `en_events.json` — нормализованные и локализованные события.

13.4.2. Сценарий для отладки правил для всех этапов обработки событий

- Чтобы настроить сценарий для отладки совместной работы правил нормализации, агрегации, обогащения и корреляции событий:

  1. Запустите файл `PTSIEMSDK_GUI.exe`.

     Откроется окно утилиты PTSIEMSDK GUI.

  2. Выберите вкладку **Сценарии**.
  3. В панели **Параметры BUILD** установите флажок **BUILD FORMULAS**.
  4. Установите флажок **BUILD AGGREGATION RULES**.
  5. Установите флажок **BUILD TABLE LISTS DATABASE**.
  6. В открывшемся блоке параметров установите флажок **Использовать файлы из раздела «Настройка»**.
  7. Установите флажок **BUILD ENRICHMENT RULES**.
  8. В открывшемся блоке параметров установите флажок **Использовать файл из раздела «Настройка»**.
  9. Установите флажок **BUILD CORRELATION RULES**.
  10. В открывшемся блоке параметров установите флажок **Использовать файл из раздела «Настройка»**.
  11. В панели **Параметры RUN** установите флажок **NORMALIZE**.
  12. В открывшемся блоке параметров справа от поля **Файл исходных событий** нажмите  и выберите файл с исходными событиями.
  13. Если исходные события не были предварительно обработаны и преобразованы в формат JSON, установите флажок **Исходные события без конвертов** и в раскрывающемся списке **MIME** выберите тип событий.
  14. Установите флажок **Показать статистику**.
  15. Установите флажок **AGGREGATE**.
  16. Установите флажок **ENRICH**.
  17. В открывшемся блоке параметров установите флажок **База данных с табличными списками**.
  18. Установите флажок **CORRELATE**.
  19. В открывшемся блоке параметров установите флажок **База данных с табличными списками**.
  20. Установите флажок **Показать статистику**.

21. В панели инструментов нажмите кнопку **Запуск**.

    Откроется окно **Сценарии**, начнется создание графов нормализации и агрегации, базы данных табличных списков, графов обогащения и корреляции, затем нормализация исходных событий, агрегация, обогащение и корреляция нормализованных событий. По окончании в окне появится статистика по событиям.

22. В окне **Сценарии** нажмите кнопку **Закрыть**.

23. В панели инструментов нажмите **Папка с результатами**.

    В открывшейся папке содержатся созданные файлы: `formulas_graph.json` — граф нормализации; `aggfilters.json` — граф агрегации; `enrules_graph.json` — граф обогащения; `corrules_graph.json` — граф корреляции; `fpta_db.db` — база данных табличных списков; `norm_events.json` — нормализованные события; `aggr_norm_events.json` — агрегированные события; `enrich_events.json` — обогащенные данными события; `corr_events.json` — корреляционные события.

14. Утилита kbtools

Утилита `kbtools` предназначена для экспорта и импорта объектов из БД Knowledge Base, создания и удаления БД. Папка `Tools` с утилитой расположена в папке установки Knowledge Base.

При запуске утилиты без ключей (или с ключами `help`, `version`) на экран выводятся версия утилиты и список ключей, позволяющих использовать утилиту для выполнения тех или иных действий. При запуске утилиты с ключом для выбора действия и ключом `help` на экран выводятся список дополнительных ключей для выполнения этого действия.

В этом разделе

14.1. exportPackage, экспорт объектов из БД

Ключ `exportPackage` позволяет использовать утилиту для экспорта всех объектов из указанной БД Knowledge Base или из набора для установки.

> **Примечание.** Для работы с объектами нужно распаковать полученный при экспорте пакет (файл с расширением `.kb`), запустив утилиту с ключом `unpack`.

Формат запуска:

```
kbtools
    exportPackage
    --db <Идентификатор БД>
    --mode DeploySet
    --deploySet <Системное название набора>
    --output <Путь к файлу с расширением .kb>
    --kbHost <IP-адрес>
    --login <Логин>
    --password <Пароль>
```

Ключи для настройки экспорта:

— `--db`

  Идентификатор БД (обязательный ключ).

— `--mode`

Варианты экспорта объектов: `All` — все объекты, `DeploySet` — объекты из набора для установки (обязательный ключ).

— `--deploySet`

Название набора для установки. Для указания вложенных наборов вы можете использовать косую черту (/).

— `--output`

Путь и название файла с расширением `.kb` для сохранения объектов (обязательный ключ).

Ключи для настройки подключения к Knowledge Base:

— `--kbHost`

Адрес узла установки Knowledge Base (обязательный ключ).

— `--kbPort`

Порт подключения к Knowledge Base (по умолчанию 8190).

— `--iamHost`

Адрес узла установки PT MC (по умолчанию совпадает с узлом установки Knowledge Base).

— `--iamPort`

Порт подключения к PT MC (по умолчанию 3334).

— `--login`

Логин для подключения к Knowledge Base (обязательный ключ).

Примечание. Создавать, изменять и удалять пользовательские БД могут только пользователи с правами администратора в Knowledge Base.

— `--password`

Пароль для подключения к Knowledge Base (обязательный ключ).

Пример для экспорта всех объектов из БД Test_DB:

```
C:\Tools\kbtools.exe exportPackage --db Test_DB --mode All --output C:
\Tools\Test_DB_content.kb --kbHost ptkb.company.ru --login <Логин> --password
<Пароль>
```

Пример для экспорта объектов из набора для установки Set_1, вложенного в набор RootSet:

```
C:\Tools\kbtools.exe exportPackage --db Test_DB --mode DeploySet --deploySet RootSet/
Set_1 --output C:\Tools\Set_1_content.kb --kbHost ptkb.company.ru --login <Логин> --
password <Пароль>
```

14.2. importPackage, импорт объектов в БД

Ключ `importPackage` позволяет использовать утилиту для импорта объектов в указанную БД Knowledge Base.

Примечание. Для импорта объектов нужно сформировать пакет (файл с расширением `.kb`), запустив утилиту с ключом `pack`.

Формат запуска:

```
kbtools
    importPackage
    --db <Идентификатор БД>
    --source <Путь к файлу с расширением .kb>
    --mode <Режим импорта>
    --kbHost <IP-адрес>
    --login <Логин>
    --password <Пароль>
```

Ключи для настройки импорта:

— `--db`

Идентификатор БД (обязательный ключ).

— `--mode`

Режим импорта объектов (обязательный ключ):

- `Upsert` — все объекты импортируются в БД как пользовательские. Новые объекты добавляются в БД, уже существующие в БД пользовательские объекты заменяются объектами из файла (в том числе — записи табличных списков).
- `UpsertOrigin` — импортируются только объекты, разработанные сторонними поставщиками. Все объекты импортируются в БД как стандартные. Новые объекты добавляются в БД, уже существующие в БД объекты этого поставщика заменяются объектами из файла.
- `ReplaceOrigin` — импортируются только объекты, разработанные сторонними поставщиками. Все объекты импортируются в БД как стандартные. Если в БД и в файле есть объекты, разработанные одним и тем же поставщиком, все такие объекты заменяются в БД объектами из файла. Таким образом, новые объекты добавляются в БД, уже существующие в БД объекты заменяются объектами из файла, отсутствующие в файле объекты удаляются из БД.

— `--source`

Путь и название файла с расширением `.kb` с объектами для импорта (обязательный ключ).

— `--ImportPtOrigin`

Импортировать объекты разработанные Positive Technologies: `true` — да, `false` — нет (по умолчанию).

— `--ImportMacros`

Импортировать макросы: `true` — да, `false` — нет (по умолчанию).

— `--ImportTaxonomy`

Импортировать схему полей событий: `true` — да, `false` — нет (по умолчанию).

Ключи для настройки подключения к Knowledge Base:

— `--kbHost`

Адрес узла установки Knowledge Base (обязательный ключ).

— `--kbPort`

Порт подключения к Knowledge Base (по умолчанию 8190).

— `--iamHost`

Адрес узла установки PT MC (по умолчанию совпадает с узлом установки Knowledge Base).

— `--iamPort`

Порт подключения к PT MC (по умолчанию 3334).

— `--login`

Логин для подключения к Knowledge Base (обязательный ключ).

Примечание. Создавать, изменять и удалять пользовательские БД могут только пользователи с правами администратора в Knowledge Base.

— `--password`

Пароль для подключения к Knowledge Base (обязательный ключ).

Пример:

```
C:\Tools\kbtools.exe importPackage --db TestRootDB --mode Upsert --source C:
\Tools\Set_1_content_mod.kb --kbHost dev-ptkb2.rd.ptsecurity.ru --login <Логин> --
password <Пароль>
```

14.3. unpack, распаковка пакета

Ключ `unpack` позволяет использовать утилиту для распаковки пакетов с объектами, экспортированных из Knowledge Base.

Формат запуска:

```
kbtools
    unpack
    --source <Путь к файлу с расширением .kb>
    --output <Путь к папке с объектами>
```

Ключи для настройки распаковки пакета:

— `--source` или `-s`

Путь и название файла с импортированными объектами (обязательный ключ).

— `--output` или `-o`

Путь к папке для сохранения объектов (обязательный ключ).

Пример:

```
C:\Tools\kbtools.exe unpack --source C:\Tools\Set_1_content.kb --output C:
\Tools\Set_1
```

14.4. pack, формирование пакета для импорта

Ключ `pack` позволяет использовать утилиту для формирования пакетов с объектами для импорта в Knowledge Base.

Формат запуска:

```
kbtools
    pack
    --source <Путь к папке с объектами>
    --output <Путь к файлу с расширением .kb>
```

Ключи для настройки формирования пакета:

— `--source` или `-s`

  Путь к папке с объектами (обязательный ключ).

— `--output` или `-o`

  Путь и название для сохранения файла для импорта (обязательный ключ).

Пример:

```
C:\Tools\kbtools.exe pack --source C:\Tools\Set_1 --output C:
\Tools\Set_1_content_mod.kb
```

14.5. createRootDB, создание БД

Ключ `createRootDB` позволяет использовать утилиту для создания в Knowledge Base пустой БД.

Формат запуска:

```
kbtools
    createRootdb
    --db <Идентификатор БД>
    --kbHost <IP-адрес>
    --login <Логин>
    --password <Пароль>
```

Ключи для настройки параметров новой БД:

— `--db`

  Идентификатор БД (обязательный ключ).

— `--db_name`

  Название БД для отображения в интерфейсе.

Ключи для настройки подключения к Knowledge Base:

— `--kbHost`

  Адрес узла установки Knowledge Base (обязательный ключ).

— `--kbPort`

  Порт подключения к Knowledge Base (по умолчанию 8190).

— `--iamHost`

  Адрес узла установки PT MC (по умолчанию совпадает с узлом установки Knowledge Base).

— `--iamPort`

  Порт подключения к PT MC (по умолчанию 3334).

— `--login`

  Логин для подключения к Knowledge Base (обязательный ключ).

  Примечание. Создавать, изменять и удалять пользовательские БД могут только пользователи с правами администратора в Knowledge Base.

— `--password`

  Пароль для подключения к Knowledge Base (обязательный ключ).

Пример:

```
C:\Tools\kbtools.exe createRootdb --db TestRootDB --db_name "Тестовая база" --kbHost
ptkb.company.ru --login <Логин> --password <Пароль>
```

14.6. dropRootDB, удаление БД

Ключ `dropRootDB` позволяет использовать утилиту для удаление БД Knowledge Base.

Формат запуска:

```
kbtools
    dropRootdb
    --db <Идентификатор БД>
    --kbHost <IP-адрес>
    --login <Логин>
    --password <Пароль>
```

Ключ для удаления БД:

— `--db`

  Идентификатор БД (обязательный ключ).

Ключи для настройки подключения к Knowledge Base:

— `--kbHost`

Адрес узла установки Knowledge Base (обязательный ключ).

— `--kbPort`

Порт подключения к Knowledge Base (по умолчанию 8190).

— `--iamHost`

Адрес узла установки PT MC (по умолчанию совпадает с узлом установки Knowledge Base).

— `--iamPort`

Порт подключения к PT MC (по умолчанию 3334).

— `--login`

Логин для подключения к Knowledge Base (обязательный ключ).

Примечание. Создавать, изменять и удалять пользовательские БД могут только пользователи с правами администратора в Knowledge Base.

— `--password`

Пароль для подключения к Knowledge Base (обязательный ключ).

Пример:

```
C:\Tools\kbtools.exe dropRootdb --db TestRootDB --kbHost ptkb.company.ru --login
<Логин> --password <Пароль>
```

15. Виды запросов к API

Раздел содержит информацию о протоколе обмена данными между сторонними приложениями и MaxPatrol SIEM. Обмен данными производится через программный интерфейс продукта (REST API) по протоколу HTTPS.

В этом разделе

15.1. Авторизация

Запрос на авторизацию и получение кода.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/connect/authorize
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 85. Параметры в теле запроса connect/authorize

Параметр	Обязательный	Тип данных	Описание
`client_id`	Да	String	Идентификатор приложения, например `mpx` для MaxPatrol SIEM, `ptkb` для Knowledge Base, `idmgr` для Management and Configuration
`client_secret`	Да	String	Ключ доступа к приложению
`redirect_uri`	Нет	String	URL для редиректа
`response_type`	Да	Array [String]	Тип ответа: — `code`; — `code id_token`; — `code id_token token`; — `code token`; — `id_token`; — `id_token token`; — `token`
`scope`	Да	Array [String]	Права доступа по токену. Доступ по API: — `<Идентификатор приложения>.api` — доступ к приложению по API (например, `mpx.api`, `ptkb.api`); — `<Привилегия>` — выдача токена только при наличии у пользователя необходимых привилегий; — `authorization` — доступ для аутентификации; — `offline_access` — доступ к приложению от имени пользователя на длительное время (при этом выдается токен для обновления). Доступ к данным: — `email` — адрес электронной почты пользователя; — `openid` — возможность использования протокола OpenID; — `phone` — телефон пользователя; — `profile` — имя и фамилия пользователя

Значение параметра `client_secret` вы можете получить в конфигурации соответствующего приложения.

Для просмотра конфигурации приложений, установленных на Linux, вы можете использовать следующие команды:

- для приложения MaxPatrol SIEM: `sudo grep ClientSecret /var/lib/deployer/role_instances/core*/params.yaml`
- для Management and Configuration: `sudo grep ClientSecret /var/lib/deployer/role_instances/ma*/params.yaml`

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 302 (Redirect) и код авторизации.

Возможные коды ошибок и их значения:

400 (Bad request) — синтаксическая ошибка в запросе.

15.2. Получение токена доступа

Запрос для получения и обновления токена доступа к приложениям, зарегистрированным в PT MC. При аутентификации используется протокол OAuth. Для получения токена доступа необходимо указать учетные данные пользователя PT MC.

Время действия токена доступа ограничено. При наличии у токена прав `offline_access` вы можете обновить его по токену для обновления (выдается вместе с токеном доступа).

Аутентификация для выполнения запроса не требуется.

Метод и URL запроса:

```
POST <Корневой URL API>:3334/connect/token
```

Тело запроса может содержать параметры, описанные в таблице ниже.

> **Внимание!** Для передачи данных в теле запроса необходимо использовать формат `application/x-www-form-urlencoded`.

Таблица 86. Параметры в теле запроса /connect/token

Параметр	Обязательный	Тип данных	Описание
`client_id`	Да	String	Идентификатор приложения, например `mpx` для MaxPatrol SIEM, `ptkb` для Knowledge Base, `idmgr` для Management and Configuration
`client_secret`	Да	String	Ключ доступа к приложению

Параметр	Обязательный	Тип данных	Описание
`grant_type`	Да	Array [String]	Тип разрешений на аутентификацию: — `authorization_code` — код доступа; — `client_credentials` — учетные данные; — `delegation` — делегирование; — `implicit` — неявное разрешение; — `password` — пароль; — `refresh_token` — токен для обновления токена доступа
`password`	Нет	String	Пароль учетной записи
`response_type`	Да	Array [String]	Тип ответа: — `code`; — `code id_token`; — `code id_token token`; — `code token`; — `id_token`; — `id_token token`; — `token`
`scope`	Да	Array [String]	Права доступа по токену. Доступ по API: — `<Идентификатор приложения>.api` — доступ к приложению по API (например, `mpx.api`, `ptkb.api`); — `<Привилегия>` — выдача токена только при наличии у пользователя необходимых привилегий; — `offline_access` — доступ к приложению от имени пользователя на длительное время (при этом выдается токен для обновления).

Параметр	Обязательный	Тип данных	Описание
			Доступ к данным: — `email` — адрес электронной почты пользователя; — `openid` — возможность использования протокола OpenID; — `phone` — телефон пользователя; — `profile` — имя и фамилия пользователя
`username`	Нет	String	Логин учетной записи пользователя PT MC
`amr`	Нет	String	Признак получения токена: `ldap`. Параметр добавляется в запрос, если токен нужно получить для учетной записи LDAP

Значение параметра `client_secret` вы можете получить в конфигурации соответствующего приложения.

Для просмотра конфигурации приложений, установленных на Linux, вы можете использовать следующие команды:

- для приложения MaxPatrol SIEM: `sudo grep ClientSecret /var/lib/deployer/role_instances/core*/params.yaml`
- для Management and Configuration: `sudo grep ClientSecret /var/lib/deployer/role_instances/ma*/params.yaml`

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 87. Поля ответа на запрос /connect/token

Поле	Тип данных	Описание
`access_token`	UUID	Токен доступа
`expires_in`	Number	Время действия токена в секундах
`id_token`	UUID	Токен текущей сессии
`refresh_token`	UUID	Токен для обновления токена доступа
`token_type`	Enum	Тип токена (Bearer)

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Пример получения токена

Запрос:

```
POST https://localhost:3334/connect/token
username:Administrator
password:P@ssw0rd
client_id:mpx
client_secret:ela74298-ff88-3268-9734-51de0a2c8d5e
grant_type:password
response_type:code id_token
scope: offline_access mpx.api ptkb.api
```

Ответ:

```
{
  "access_token": "999bcb24f71e0090bb10fab3ef99705f79dece3b41a538f3be967093e89d5c3e",
  "expires_in": 86400,
  "token_type": "Bearer",
  "refresh_token": "68f2c0f5d0b37ad267a2c828dfe8c1ef7e012b7f83a0b47a58a117e99d41478f"
}
```

Пример обновления токена

Запрос:

```
POST https://localhost:3334/connect/token
username:Administrator
password:P@ssw0rd
client_id:mpx
client_secret:ela74298-ff88-3268-9734-51de0a2c8d5e
grant_type:refresh_token
response_type:code id_token
refresh_token:68f2c0f5d0b37ad267a2c828dfe8c1ef7e012b7f83a0b47a58a117e99d41478f
```

Ответ:

```
{
  "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6ImdGcHZFcW1fS1hpOHh4alhHTnQtbnlhZUxWTSIsInR5cCI6IkpXVCIs
Ing1dCI6ImdGcHZFcW1fS1hpOHh4alhHTnQtbnlhZUxWTSJ9.eyJuYmYiOjE2MjY2ODIzOTQsImV4cCI6MTYy
NjY4MjY5NCwiaXNzIjoiaHR0cHM6Ly9tcHgtdHcyLnJkLnB0c2VjdXJpdHkucnU6MzMzNCIsImF1ZCI6Im1we
CIsImlhdCI6MTYyNjY4MjM5NCwiYXRfaGFzaCI6Ik5oRlFIektIeTFUTEo0UWFzUUZDMmciLCJzdWIiOiJiY2
ZhYzhhYy0zYzBlLTQwZDUtYTEzZS01NjI1MmI0ZTJkZDciLCJhdXRoX3RpbWUiOjE2MjY2ODIzNzcsImlkcCI
6ImxvY2FsIiwiYW1yIjpbInBhc3N3b3JkIl19.brh26LOPFG2GQ-
gX1RA4nvwdtaXdx3l48SrD8pmz67xjaoaQCAxLkAEXmHYi181ovaHJYVh49jt1gaPWyaAWMuEiLJVQcVsQauy
fyo3h0lRTMKpGPK203XBsmfF18UHw4rgHbEKtFBsNx_VTC00QVUlDsczOZ9oq3vJXyfkQJ6Sk8SDU71EyFv_K
```

```
sz1LeNLaqK93GmeKj5P3BMU0sH4-5_lEU2jOtgPT1z4CuPnOprAzwFb2_4-
T1CCkw2FQn7_KERzmH7IxcLxcoD-
WbZSVeayjlmjDHrbDteh7LezDxem-8WUpFOpRJ8RH__X0qmd00Xl3f5YNNYqoDAzbxzx0bA",
  "access_token": "db13c5e795cdd9f43366ea12101cd9432f65c3800fbff4d0c2876aa630b9b27f",
  "expires_in": 86400,
  "token_type": "Bearer",
  "refresh_token": "139c8b94705bf596cbe14d536268e70e92b2cf600fa5da2dc71349da6d4df2b1"
}
```


15.3. Работа с группами активов

С помощью запросов к API вы можете получить идентификаторы созданных в MaxPatrol SIEM пользовательских инфраструктур и групп активов. Также вы можете создавать и удалять группы активов.

В этом разделе

Получение идентификатора инфраструктуры (см. раздел 15.3.1)

Получение идентификатора группы активов (см. раздел 15.3.2)

Создание группы активов (см. раздел 15.3.3)

Проверка статуса создания группы (см. раздел 15.3.4)

Удаление группы активов (см. раздел 15.3.5)

15.3.1. Получение идентификатора инфраструктуры

Запрос для получения идентификаторов созданных инфраструктур.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scopes/v2/scopes
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 88. Параметры в теле запроса /api/scopes/v2/scopes

Параметр	Обязательный	Тип данных	Описание
`limit`	Нет	Number	Максимальное количество элементов инфраструктуры в ответе
`offset`	Нет	Number	Количество отбрасываемых элементов

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 89. Поля ответа на запрос /api/scopes/v2/scopes

Поле	Тип данных	Описание
`id`	UUID	Идентификатор инфраструктуры
`name`	String	Название инфраструктуры
`tenantId`	UUID	Идентификатор дочерней организации

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/scopes/v2/scopes
```

Ответ:

```
[
  {
    "id": "00000000-0000-0000-0000-000000000005",
    "name": "Инфраструктура по умолчанию",
    "tenantId": "32b17948-5755-4905-b0f3-17db2402256a"
  }
]
```

15.3.2. Получение идентификатора группы активов

Запрос для получения идентификаторов групп активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_temporal_readmodel/v2/groups/hierarchy
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 90. Параметры в теле запроса /api/assets_temporal_readmodel/v2/groups/hierarchy

Параметр	Обязательный	Тип данных	Описание
`type`	Нет	Enum	Тип группы: — `all` — любой; — `static` — статическая; — `dynamic` — динамическая
`groupId`	Нет	UUID	Идентификатор группы

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 91. Поля ответа на запрос /api/assets_temporal_readmodel/v2/groups/hierarchy

Поле	Тип данных	Описание
`id`	UUID	Идентификатор группы
`name`	String	Имя группы для отображения в интерфейсе
`groupType`	Enum	Тип группы: — `all` — любой; — `static` — статическая; — `dynamic` — динамическая
`isReadOnly`	Bool	Возможно ли изменение группы
`isRemovable`	Bool	Возможно ли удаление группы
`isRoot`	Bool	Является ли группа корневой
`isInvalidPredicate`	Bool	—
`isSlow`	Bool	—
`treePath`	String	Полный путь к группе в списке
`children`	Array	Список вложенных групп

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/assets_temporal_readmodel/v2/groups/hierarchy
```

Ответ:


```
[
  {
    "id": "00000000-0000-0000-0000-000000000002",
    "name": "Root",
    "groupType": "static",
    "isReadOnly": true,
    "isRemovable": false,
    "isRoot": true,
    "isInvalidPredicate": false,
    "isSlow": false,
    "treePath": "Root",
    "children": [
      {
        "id": "1685ce7c-e440-0001-0000-000000000005",
        "name": "Import",
        "groupType": "static",
        "isReadOnly": false,
        "isRemovable": true,
        "isRoot": false,
        "isInvalidPredicate": false,
        "isSlow": false,
        "treePath": "Import",
        "children": []
      },
      {
        "id": "1685cf07-70c0-0001-0000-000000000009",
        "name": "Windows",
        "groupType": "dynamic",
        "isReadOnly": false,
        "isRemovable": true,
        "isRoot": false,
        "isInvalidPredicate": false,
        "isSlow": false,
        "treePath": "Windows",
        "children": []
      },
      {
        "id": "00000000-0000-0000-0000-000000000003",
        "name": "Unmanaged hosts",
        "groupType": "dynamic",
```

```
        "isReadOnly": true,
        "isRemovable": false,
        "isRoot": false,
        "isInvalidPredicate": false,
        "isSlow": false,
        "treePath": "Unmanaged hosts",
        "children": []
      }
    ]
  }
]
```


15.3.3. Создание группы активов

Запрос для создания группы активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v2/groups
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 92. Параметры в теле запроса api/assets_processing/v2/groups

Параметр	Обязательный	Тип данных	Описание
`description`	Нет	String	Описание группы
`groupType`	Да	Enum	Тип группы: — `undefined` — не определен; — `static` — статическая; — `dynamic` — динамическая
`metrics`	Да	Array [String]	Контекстные метрики CVSS: — `ar` — требования к доступности; — `cdp` — вероятность нанесения косвенного ущерба; — `cr` — требования к конфиденциальности; — `ir` — требования к целостности; — `td` — плотность целей
`name`	Да	String	Название группы

Параметр	Обязательный	Тип данных	Описание
`organizationInformation`	Да	Array [String]	Информация об организации: — `contactUserId` — идентификатор пользователя, являющегося контактным лицом; — `address` — адрес
`organizationInfrastructure`	Да	Array [String]	Информация об инфраструктуре: — `internetProviders` — провайдеры интернета; — `numberOfNodes` — количество узлов на периметре; — `registeredDomains` — зарегистрированные домены; — `usedNetworkApplications` — используемые веб-приложения; — `usedNetworks` — используемые сети
`parentId`	Да	UUID	Идентификатор родительской группы
`predicate`	Нет	String	PDQL-фильтр активов для динамической группы

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 93. Поля ответа на запрос api/assets_processing/v2/groups

Поле	Тип данных	Описание
`operationId`	UUID	Идентификатор операции создания группы

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/assets_processing/v2/groups
{
```

```
    "name": "Import",
    "description": "Import",
    "parentId": "00000000-0000-0000-0000-000000000002",
    "groupType": "static",
    "metrics": {
      "ar": "ND",
      "cdp": "ND",
      "cr": "ND",
      "ir": "ND",
      "td": "ND"
    },
    "organizationInformation": {
      "address": "",
      "contactUserId": ""
    },
    "organizationInfrastructure": {
      "internetProviders": "",
      "numberOfNodes": "",
      "registeredDomains": "",
      "usedNetworkApplications": "",
      "usedNetworks": ""
    }
  }
```


Ответ:


```
  {
    "operationId": "14a5ca6b-f780-a001-0000-00000000003f"
  }
```


15.3.4. Проверка статуса создания группы

Запрос для получения статуса операции по созданию группы активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v2/groups/operations/<Идентификатор операции>
```

URL запроса должен содержать идентификатор операции по созданию группы активов.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 94. Поля ответа на запрос api/assets_processing/v2/groups/operations/<Идентификатор операции>

Поле	Тип данных	Описание
`<Идентификатор группы>`	UUID	Идентификатор группы

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция по созданию группы выполняется.

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/assets_processing/v2/groups/operations/168600a8-8f00-a001-0000-000000000008
```

Ответ:

```
"168600a9-8dc0-0001-0000-00000000000d"
```

15.3.5. Удаление группы активов

Запрос для удаления группы активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v2/groups/removeOperation
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 95. Параметры в теле запроса /api/assets_processing/v2/groups/removeOperation

Параметр	Обязательный	Тип данных	Описание
`groupIds`	Да	Array [UUID]	Идентификаторы групп

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 96. Поля ответа на запрос /api/assets_processing/v2/groups/removeOperation

Поле	Тип данных	Описание
`operationId`	UUID	Идентификатор операции удаления группы

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/assets_processing/v2/groups/removeOperation
{
  "groupIds": [
    "14a57a85-8b00-0001-0000-000000000015"
  ]
}
```

Ответ:

```
{
  "operationId": "14cdba4b-c680-a001-0000-00000000384e"
}
```

15.4. Импорт активов из CSV

С помощью запросов к API вы можете импортировать в MaxPatrol SIEM список активов из CSV-файла. Для этого необходимо:

1. Если в модель активов были добавлены пользовательские поля — получить данные этих полей.
2. Создать CSV-файл в соответствии с требуемым форматом.
3. Загрузить CSV-файл со списком активов.
4. Запустить процедуру импорта активов.

В этом разделе

Получение пользовательских полей (см. раздел 15.4.1)

Запрос формата CSV-файла (см. раздел 15.4.2)

Загрузка CSV-файла (см. раздел 15.4.3)

Выгрузка файла ошибок (см. раздел 15.4.4)

Запуск импорта активов (см. раздел 15.4.5)

Проверка статуса импорта (см. раздел 15.4.6)

15.4.1. Получение пользовательских полей

Запрос для получения данных пользовательских полей в модели активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v2/csv/metadata
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 97. Поля ответа на запрос /api/assets_processing/v2/csv/metadata

Поле	Тип	Описание
`modelTypeName`	String	Название корневой группы параметров в пользовательской модели активов
`propertyDeclarations`	Array	Свойства пользовательского поля
`propertyName`	String	Название пользовательского поля
`propertyType`	String	Тип данных пользовательского поля

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/assets_processing/v2/csv/metadata
```

Ответ:

```
[
  {
    "propertyName": "uf_assetnumber",
    "propertyDeclarations": [
```

```
        {
          "modelTypeName": "Core.Host",
          "propertyType": "int"
        }
      ]
    },
    {
      "propertyName": "uf_assetowner",
      "propertyDeclarations": [
        {
          "modelTypeName": "Core.Host",
          "propertyType": "string"
        }
      ]
    },
    {
      "propertyName": "uf_assetrevisiondate",
      "propertyDeclarations": [
        {
          "modelTypeName": "Core.Host",
          "propertyType": "datetime"
        }
      ]
    }
  ]
```


15.4.2. Запрос формата CSV-файла

Запрос для получения формата CSV-файла.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v2/csv/example
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Тело ответа содержит пример содержимого CSV-файла с данными актива.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/assets_processing/v2/csv/example
```

Ответ:

```
"typealias";"Fqdn";"Hostname";"Ip";"Mac";"IsVirtual"
"";"dns.somedomain.ru";"w2k3sp2x86s5";"192.168.0.4|182.168.10.1";"00:50:56:A6:0C:36|
00:60:56:A6:0C:36";"false"
```

15.4.3. Загрузка CSV-файла

Запрос для загрузки CSV-файла с данными активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v2/csv/import_operation
```

Параметры строки запроса описаны в таблице ниже.

Таблица 98. Параметры строки запроса /api/assets_processing/v2/csv/import_operation

Параметр	Обязательный	Тип данных	Описание
`scope`	Да	UUID	Идентификатор инфраструктуры

В запрос необходимо добавить заголовок `Content-Disposition` с описанием CSV-файла, а в тело запроса — CSV-файл с данными активов.

Формат CSV-файла

Файл должен быть представлен в кодировке UTF-8 с BOM. Первая строка файла должна содержать названия полей актива. Вторая и последующие строки — значения полей импортируемых активов (одна строка соответствует одному активу). Значения полей должны быть разделены точкой с запятой. Значения текстовых полей должны быть заключены в кавычки.

Таблица 99. Колонки CSV-файла

Колонка	Тип данных	Описание
`Fqdn`	String	Полное доменное имя
`Hostname`	String	Имя узла
`Ip`	Array [String]	IP-адрес. Поле может содержать несколько значений, которые должны быть разделены вертикальной чертой

Колонка	Тип данных	Описание
IsVirtual	Bool	Является ли виртуальным
Mac	String [String]	MAC-адрес. Поле может содержать несколько значений, которые должны быть разделены вертикальной чертой
typealias	Enum	Тип актива: — Alcatel OmniSwitch (под управлением AOS) — omniswitch; — BSD, FreeBSD и macOS — bsd; — Check Point GAiA OS (межсетевой экран) — checkpoint; — Check Point SPLAT (межсетевой экран) — checkpoint; — Cisco ACS — acs; — Cisco ASA (межсетевой экран) — asa; — Cisco FWSM (межсетевой экран) — fwsm; — Cisco IOS — ios; — Cisco IOS XE — ios; — Cisco ISE — ise; — Cisco Nexus — nexus; — Cisco PIX (межсетевой экран) — pix; — FortiNet FortiGate (межсетевой экран) — fortigate; — HPE HP-UX — hp_ux; — Huawei VRP — vrp; — IBM AIX — aix; — Juniper JunOS — junos; — Linux (все семейство ОС) — linux; — Oracle Solaris — solaris; — Palo Alto Networks PAN-OS (межсетевые экраны) — pan_os; — VMware vSphere Hypervisor (ESXi) — esxi; — Windows — windows

Колонка	Тип данных	Описание
<Пользователь-ские поля>	—	Добавленные пользователем поля актива, не являющиеся стандартными

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 100. Поля ответа на запрос /api/assets_processing/v2/csv/import_operation

Поле	Тип данных	Описание
id	UUID	Идентификатор операции импорта
isLogFileCreated	Bool	Создан ли файл с ошибками
rowsCountExceeded	Bool	Превышает ли количество строк в импортируемом файле установленное ограничение
totalRowsCount	Number	Общее количество строк в импортируемом файле
validRowsCount	Number	Количество строк с валидными данными активов в импортируемом файле

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/assets_processing/v2/csv/import_operation?
scopeId=00000000-0000-0000-0000-000000000005
```

Заголовок запроса:

```
Content-Disposition:form-data; name="upfile"; filename="Assets_list.csv"
```

В тело запроса (в представлении `form-data`) необходимо добавить CSV-файл `Assets_list.csv` с ключом `upfile`.

Ответ:

```
{
  "id": "0e372968-e18b-471e-b3b7-ad0d927c2dd9",
  "isLogFileCreated": true,
  "rowsCountExceeded": false,
  "validRowsCount": 2,
```

```
  "totalRowsCount": 3
}
```

15.4.4. Выгрузка файла ошибок

Запрос для выгрузки списка ошибок, обнаруженных при импорте CSV-файла со списком активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v2/csv/import_operation/<Идентификатор операции>/logfile
```

URL запроса должен содержать идентификатор операции импорта CSV-файла.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Заголовок ответа `Content-Disposition` содержит информацию о выгруженном файле с ошибками. Тело ответа содержит список ошибок с указанием номера строки в импортированном CSV-файле. Если файл с ошибками не создан, сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost/api/assets_processing/v2/csv/import_operation/0e372968-e18b-471e-b3b7-ad0d927c2dd9/logfile
```

Ответ:

```
"3";"Значение [version] некорректно для псевдонима"
```

15.4.5. Запуск импорта активов

Запрос для запуска импорта активов из загруженного CSV-файла.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v2/csv/import_operation/<Идентификатор операции>/start
```

URL запроса должен содержать идентификатор операции импорта CSV-файла.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 101. Параметры в теле запроса api/assets_processing/v2/csv/import_operation/<Идентификатор операции>/start

Параметр	Обязательный	Тип данных	Описание
groupsId	Да	Array [UUID]	Идентификаторы групп, в которые нужно добавить активы

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/assets_processing/v2/csv/import_operation/0e372968-e18b-471e-b3b7-ad0d927c2dd9/start
{
  "groupsId": [
    "1499141b-4c00-0001-0000-00000000000d"
  ]
}
```

15.4.6. Проверка статуса импорта

Запрос для получения статуса импорта активов из CSV-файла.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v2/csv/import_operation/<Идентификатор операции>/state
```

URL запроса должен содержать идентификатор операции импорта CSV-файла.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 102. Поля ответа на запрос api/assets_processing/v2/csv/import_operation/<Идентификатор файла>/state

Поле	Тип данных	Описание
`errorModel`	Array	Данные ошибок
`errorModel` → `code`	Number (int32)	Код ошибки
`errorModel` → `errors`	Array	Название ошибки
`errorModel` → `message`	String	Текст ошибки
`state`	Enum	Статус импорта: — `completed` — завершен; — `failed` — не выполнен; — `inprogress` — выполняется; — `validated` — на проверке
`succeedCount`	Number	Количество импортированных активов
`updatedGroups`	Array	Данные обновленной группы
`updatedGroups` → `displayName`	String	Название группы
`updatedGroups` → `id`	UUID	Идентификатор группы

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/assets_processing/v2/csv/import_operation/0e372968-e18b-471e-b3b7-ad0d927c2dd9/state
```

Ответ:


```
{
  "state": "completed",
  "succeedCount": 1,
  "updatedGroups": [
    {
      "id": "149f37fa-c2c0-0001-0000-000000000011",
      "displayName": "Import"
    }
  ],
  "errorModel": null
}
```


15.5. Работа с табличными списками в MaxPatrol SIEM

С помощью запросов к API вы можете экспортировать и импортировать записи табличных списков, установленных в MaxPatrol SIEM, обновлять, очищать, искать табличные списки, а также получать информацию о табличном списке.

В этом разделе

15.5.1. Поиск табличного списка

Запрос для поиска табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/events/v2/table_lists
```

Параметры строки запроса описаны в таблице ниже.

Таблица 103. Параметры строки запроса /api/events/v2/table_lists

Параметр	Обязательный	Тип данных	Описание
`kind`	Нет	Enum	Назначение табличного списка: — `registry` — справочник; — `correlationRule` — для правил корреляции; — `enrichmentRule` — для правил обогащения; — `assetGrid` — для данных об активах; — `cybsiGrid` — репутационный табличный список
`siemId`	Нет	String	Идентификатор конвейера обработки событий

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 104. Поля ответа на запрос /api/events/v2/table_lists

Поле	Тип данных	Описание
`name`	String	Название
`editable`	Bool	Возможно ли изменение пользователем
`token`	String	Токен доступа к списку
`ttlEnabled`	Bool	Ограничено ли время жизни записей
`fillType`	Enum	Назначение
`notifications`	Array [String]	Список ошибок

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации;

— 503 (SIEM недоступен) — ошибка доступа к конвейеру обработки событий.

Пример

Запрос:

```
GET https://localhost/api/events/v2/table_lists?kind=assetGrid
```

Ответ:

```
[
  {
    "name": "AssetGrid_AD_Domain_Controllers",
    "editable": false,
    "token": "8ea181098d4f6225750c98b17d110d5e68664c753573d6c2c24c9c44fc606c06",
    "ttlEnabled": false,
    "fillType": "assetGrid",
    "notifications": null
  },
  {
    "name": "Critical_hosts",
    "editable": false,
    "token": "65f39698a6c5a75bf14d061089926f897f4de2740e8855930b13736798a15460",
    "ttlEnabled": false,
    "fillType": "assetGrid",
    "notifications": null
  }
]
```

15.5.2. Получение информации о табличном списке

Запрос для получения информации о табличном списке.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/events/v2/table_lists/<Токен доступа>
```

URL запроса должен содержать токен доступа к табличному списку.

Параметры строки запроса описаны в таблице ниже.

Таблица 105. Параметры строки запроса /api/events/v2/table_lists/<Токен доступа>

Параметр	Обязательный	Тип данных	Описание
`siemId`	Нет	String	Идентификатор конвейера обработки событий

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 106. Поля ответа на запрос /api/events/v2/table_lists/<Токен доступа>

Поле	Тип данных	Описание
Поля с информацией о табличном списке		
`type`	Enum	Назначение
`name`	String	Название
`editable`	Bool	Возможно ли изменение пользователем
`description`	String	Описание
`notifications`	Array	Список ошибок
`created`	DateTime	Дата и время создания
`lastUpdated`	DateTime	Дата и время последнего изменения
`currentSize`	Number	Количество записей
`fields`	Array	Описание полей
Параметры описания полей (fields)		
`name`	String	Название
`type`	Enum	Тип данных: — `datetime` — дата и время; — `number` — число; — `string` — строка; — `regex` — регулярное выражение (только для табличного списка типа «справочник»)
`index`	Bool	Является ли индексируемой
`primaryKey`	Bool	Является ли ключевой
`nullable`	Bool	Может ли содержать null

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации;

— 503 (SIEM недоступен) — ошибка доступа к конвейеру обработки событий.

Пример

Запрос:

```
GET https://localhost/api/events/v2/table_lists/
0f3b138df02dd518f7a11f0bd7b85dde6f408c6e41b959838ed1ee7d2c9bbdc2
```

Ответ:


```
{
  "type": "Registry",
  "name": "Allowed_firefox_settings_accessers",
  "editable": false,
  "description": "Исполняемые файлы, которым разрешен доступ к параметрам браузера
Mozilla Firefox",
  "created": "2021-03-15T17:17:25.0000000Z",
  "lastUpdated": "2021-04-30T08:55:35.0000000Z",
  "currentSize": 2,
  "fields": [
    {
      "name": "_id",
      "type": "number",
      "index": true,
      "primaryKey": false,
      "nullable": true
    },
    {
      "name": "_last_changed",
      "type": "datetime",
      "index": true,
      "primaryKey": false,
      "nullable": true
    },
    {
      "name": "filepath",
      "type": "string",
      "index": true,
      "primaryKey": true,
      "nullable": false
    }
  ]
}
```


15.5.3. Экспорт табличного списка

Запрос для экспорта записей из табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/events/v1/table_lists/<Название>/export
```

URL запроса должен содержать название табличного списка.

Параметры строки запроса описаны в таблице ниже.

Таблица 107. Параметры строки запроса /api/events/v1/table_lists/<Название>/export

Параметр	Обязательный	Тип данных	Описание
`siemId`	Нет	String	Идентификатор конвейера обработки событий

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 108. Параметры в теле запроса api/events/v1/table_lists/<Название>/export

Параметр	Обязательный	Тип данных	Описание
`filter`	Да	String	PDQL-запрос
`filter` → `select`	Да	Array [String]	Предикат PDQL-запроса
`filter` → `where`	Нет	String	Предикат PDQL-запроса
`filter` → `timeZone`	Нет	Number	Часовой пояс в минутах
`limit`	Нет	Number	Максимальное количество записей в ответе (по умолчанию 50)

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Тело ответа содержит CSV-файл с записями табличного списка.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/events/v1/table_lists/Allowed_firefox_settings_accessers/
export
{
  "filter": {
```

```
      "select": [
        "_id",
        "_last_changed",
        "filepath"
      ],
      "where": "",
      "timeZone": 180
    }
  }
```

Ответ:

```
"_id";"_last_changed";"filepath"
"1";"30.04.2021 08:55:35";"c:\program files (x86)\mozilla firefox\firefox.exe"
"0";"30.04.2021 08:55:35";"c:\program files\mozilla firefox\firefox.exe"
```

15.5.4. Импорт табличного списка

Запрос для импорта записей в табличный список.

> **Внимание!** Вы можете импортировать записи только в табличные списки для правил обогащения или корреляции.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/events/v1/table_lists/<Название>/import
```

URL запроса должен содержать название табличного списка.

Параметры строки запроса описаны в таблице ниже.

Таблица 109. Параметры строки запроса /api/events/v1/table_lists/<Название>/import

Параметр	Обязательный	Тип данных	Описание
`update`	Нет	Bool	Признак выполнения функции upsert
`siemId`	Нет	String	Идентификатор конвейера обработки событий

В заголовке запроса `Content-Type` необходимо указать тип данных `text/csv; charset=utf-8`.

В теле запроса (в представлении `row text`) необходимо указать записи для импорта. Первая строка должна содержать названия колонок, вторая и последующие строки — значения колонок. Одна строка должна соответствовать одной записи. Названия и значения колонок должны быть заключены в кавычки и разделены точкой с запятой.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 110. Поля ответа на запрос api/events/v1/table_lists/<Название>/import

Поле	Тип данных	Описание
recordsNum	Number	Общее количество записей
importedNum	Number	Количество импортированных записей
badRecordsNum	Number	Количество неимпортированных записей
skippedRecordsNum	Number	Количество записей, отброшенных из-за превышения установленного для табличного списка максимального количества записей

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/events/v1/table_lists/VPN_Sessions_PaloAlto/import
Content-Type: text/csv; charset=utf-8
"user";"active_sessions"
"user1";"8"
"user2";"14"
"user3";"2"
```

Ответ:

```
{
  "recordsNum": 3,
  "importedNum": 3,
  "badRecordsNum": 0,
  "skippedRecordsNum": 0
}
```

15.5.5. Очистка табличных списков

Запрос для очистки табличных списков.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/events/v2/table_lists/<Токен доступа>/content
```

URL запроса должен содержать токен доступа к табличному списку.

Параметры строки запроса описаны в таблице ниже.

Таблица 111. Параметры строки запроса /api/events/v2/table_lists/<Токен доступа>/content

Параметр	Обязательный	Тип данных	Описание
`siemId`	Нет	String	Идентификатор конвейера обработки событий

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 112. Поля ответа на запрос /api/events/v2/table_lists/<Токен доступа>/content

Поле	Тип данных	Описание
`result`	ClearResult	Результат операции очистки данных таблицы

Возможные коды ошибок и их значения:

— 400 (Некорректный параметр) — синтаксическая ошибка в запросе;

— 401 (Ошибка доступа) — ошибка аутентификации;

— 503 (SIEM недоступен) — ошибка доступа к конвейеру обработки событий.

15.5.6. Обновление табличного списка

Запрос для обновления табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/events/v2/table_lists/<Токен доступа>/content
```

URL запроса должен содержать токен доступа к табличному списку.

Параметры строки запроса описаны в таблице ниже.

Таблица 113. Параметры строки запроса /api/events/v2/table_lists/<Токен доступа>/content

Параметр	Обязательный	Тип данных	Описание
`siemId`	Нет	String	Идентификатор конвейера обработки событий

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 114. Параметры в теле запроса api/events/v2/table_lists/<Токен доступа>/content

Параметр	Обязательный	Тип данных	Описание
`payload`	Да	UpdateTableContentPayload	Команды
Параметры команды			
`add`	Нет	Object	Добавленные записи. Каждая строка представлена массивом значений столбцов в порядке, объявленном в схеме
`remove`	Нет	Object	Удаленные записи. Каждая строка представлена массивом значений столбцов в порядке, объявленном в схеме

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 115. Поля ответа на запрос /api/events/v2/table_lists/<Токен доступа>/content

Поле	Тип данных	Описание
`result`	UpdateResult	Результат операции обновления данных таблицы

Возможные коды ошибок и их значения:

- 400 (Некорректный параметр) — синтаксическая ошибка в запросе;
- 401(Ошибка доступа) — ошибка аутентификации;
- 503 (Возвращается в случае реинициализации SIEM при обновлении данных из Knowledge Base) — ошибка доступа.

15.6. Работа с табличными списками в Knowledge Base

С помощью запросов к API вы можете создавать табличные списки, экспортировать и импортировать записи табличных списков. Также вы можете устанавливать табличные списки в MaxPatrol SIEM.

Для импорта записей в табличный список необходимо:

1. Открыть сессию импорта записей.
2. Загрузить CSV-файл с записями.
3. Запустить импорт записей.

В этом разделе

15.6.1. Создание табличного списка

Запрос для создания табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/tabular-lists
```

Параметры строки запроса описаны в таблице ниже.

Таблица 116. Параметры строки запроса /api-studio/siem/tabular-lists

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 117. Параметры в теле запроса /api-studio/siem/tabular-lists

Параметр	Обязательный	Тип данных	Описание
Параметры табличного списка			
`assetGroupsToSave`	Нет	Array [UUID]	—
`description`	Нет	String	Описание
`fields`	Да	Array	Параметры колонок (см. ниже)
`fillType`	Да	Enum (String)	Назначение: — `registry` — справочник; — `correlationRule` — для правил корреляции; — `enrichmentRule` — для правил обогащения; — `assetGrid` — для данных об активах; — `cybsiGrid` — репутационный табличный список
`folderId`	Нет	UUID	Список системных идентификаторов групп активов. PDQL-запрос выполняется к данным активов в указанных группах. Для табличного списка с данными об активах
`groupsToSave`	Нет	Array [UUID]	—
`includeNested`	Нет	Bool	Выполнять ли PDQL-запрос к данным активов во вложенных группах (по умолчанию `false`). Для табличного списка с данными об активах
`itemType`	Да	String	Параметр для выбора объектов репутационного списка: `ip` — IP-адрес, `domain` — имя домена, `hash` — значение контрольной суммы. Для репутационного табличного списка
`maxSize`	Да	Number (int32)	Максимальное число записей. Для табличного списка для правил корреляции и обогащения
`minScore`	Да	Number (int32)	Параметр для ввода значения интегральной уязвимости (целое число от 0 до 100), при превышении которого объект попадет в табличный список. Если параметр не указан, в табличный список попадут все объекты. Для репутационного табличного списка

Параметр	Обязательный	Тип данных	Описание
pdqlQuery	Да	String	PDQL-запрос для заполнения табличного списка. Для табличного списка с данными об активах
systemName	Да	String	Название
ttl	Да	Number (int32)	Время жизни записи табличного списка в секундах; 0 — время жизни не ограничено. Для табличного списка для правил корреляции и обогащения
ttlEnabled	Нет	Bool	Ограничено ли время жизни записи табличного списка. Для табличного списка для правил корреляции и обогащения
typicalSize	Да	Number (int32)	Рекомендуемое число записей. Для табличного списка для правил корреляции и обогащения
userCanEditContent	Да	Bool	Может ли пользователь добавлять, изменять и удалять записи в табличном списке
Параметры колонок (fields)			
isIndex	Нет	Bool	Является ли индексируемой
isNullable	Нет	Bool	Может ли содержать null
isPrimaryKey	Да	Bool	Является ли ключевой
mapping	Нет	String	—
name	Да	String	Название
typeId	Да	Number (int32)	Тип данных

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 118. Поля ответа на запрос /api-studio/siem/tabular-lists

Поле	Тип данных	Описание
id	UUID	Идентификатор объекта БД

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/tabular-lists?
contentDatabase=siem_content_for_test
{
  "systemName": "TableLiat_Example",
  "description": {},
  "typicalSize": 80000,
  "maxSize": 100000,
  "ttl": 86400,
  "ttlEnabled": true,
  "minScore": 70,
  "itemType": "Ip",
  "includeNested": true,
  "fillType": "Registry",
  "userCanEditContent": false,
  "groupsToSave": [],
  "assetGroupsToSave": [],
  "fields": [
    {
      "name": "First_Column",
      "typeId": 1,
      "isPrimaryKey": true,
      "isIndex": true
    },
    {
      "name": "Second_Column",
      "typeId": 2,
      "isPrimaryKey": false,
      "isNullable": true
    }
  ]
}
```

Ответ:

```
{
  "Id": "3fd59caf-80eb-4963-b0ca-4ab87b19b988"
}
```

15.6.2. Поиск объекта БД

Запрос для поиска объекта в БД.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/objects/list
```

Параметры строки запроса описаны в таблице ниже.

Таблица 119. Параметры строки запроса /api-studio/siem/objects/list

Параметр	Обязательный	Тип данных	Описание
contentDatabase	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 120. Параметры в теле запроса /api-studio/siem/objects/list

Параметр	Обязательный	Тип данных	Описание
filters	Да	Array	Параметры фильтра
folderId	Да	UUID	—
groupId	Да	UUID	—
recursive	Да	Bool	—
search	Да	String	Значение для поиска
skip	Да	Number (int32)	—
sort	Да	Array	Параметры сортировки (см. ниже)
take	Да	Number (int32)	—
withoutGroups	Да	Bool	—
Параметры сортировки (sort)			
name	Да	String	—
order	Да	Number (int32)	—
type	Да	Number (int32)	—

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 121. Поля ответа на запрос /api-studio/siem/objects/list

Поле	Тип данных	Описание
Count	Number	Количество найденных объектов
Rows	Array	Параметры объектов БД
Поля объектов БД		
Rows → Actions	Array	—
Rows → Actions → CanDelete	Bool	—
Rows → Actions → CanEdit	Bool	—
Rows → CompilationStatus	Array	—
Rows → CompilationStatus → CompilationStatusId	Number (int32)	—
Rows → CompilationStatus → SdkVersion	String	—
Rows → CompilationStatus → SdkVersionGreater	Bool	—
Rows → CompilationStatus → SdkVersionLess	Boolean	—
Rows → CompilationStatus → SiemProductVersion	String	—
Rows → CompilationStatus → TaxonomyVersion	String	—

Поле	Тип данных	Описание
`Rows` → `CopyOf` →	Array	Данные об исходном объекте, если объект является копией
`Rows` → `CopyOf` → `Id`	UUID	Идентификатор
`Rows` → `CopyOf` → `Name`	String	Название
`Rows` → `DeploymentStatus`	String	Статус установки в MaxPatrol SIEM: — `NotInstalled` — не установлен; — `Installed` — установлен; — `Outdated` — устарел
`Rows` → `Description`	String	Описание
`Rows` → `FolderId`	UUID	Идентификатор папки с объектом
`Rows` → `FolderPath`	String	Путь к папке с объектом
`Rows` → `id`	UUID	Идентификатор
`Rows` → `ObjectId`	String	Идентификатор Knowledge Base
`Rows` → `ObjectKind`	Enum (String)	Тип объекта БД: — `AggregationRule` — правило агрегации; — `CorrelationRule` — правило корреляции; — `EnrichmentRule` — правило обогащения; — `NormalizationRule` — правило нормализации; — `TabularList` — табличный список
`Rows` → `Origin` →	Array	—
`Rows` → `Origin` → `Id`	UUID	—
`Rows` → `Origin` → `Name`	String	—
`Rows` → `Origin` → `Type`	String	—
`Rows` → `OriginId`	UUID	Исходный идентификатор
`Rows` → `SystemName`	String	Название

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/objects/list?
contentDatabase=siem_content_for_test
{
  "search": "Auditd_processes_with_system_API_whitelist",
  "folderId": null,
  "recursive": true,
  "groupId": null,
  "withoutGroups": false,
  "skip": 0,
  "take": 50,
  "filters": {
    "SiemObjectType": [
      "TabularList"
    ]
  },
  "sort": [
    {
      "name": "objectId",
      "order": 0,
      "type": 0
    }
  ]
}
```

Ответ:

```
{
  "Rows": [
    {
      "Id": "e157f9c4-a67e-49ca-96d3-943fcbcfb026",
      "ObjectId": "PT-TL-150",
      "SystemName": "Auditd_processes_with_system_API_whitelist",
      "Description": "Исключения для правила
\"Detect_system_API_calls_from_suspicious_dir\"",
      "ObjectKind": "TabularList",
      "FolderId": "039818f1-4606-41e6-a811-da435606a604",
      "FolderPath": "Linux. Подозрительные действия пользователей\\tabular_lists",
      "OriginId": "95a1cca9-40b5-4dae-98a2-26aa36948c3c",
      "Origin": {
```

```
        "Type": "system",
        "Name": "Positive Technologies",
        "Id": "00000000-0000-0000-0000-000000000000"
      },
      "CopyOf": null,
      "CompilationStatus": {
        "TaxonomyVersion": null,
        "SdkVersion": null,
        "SiemProductVersion": "24.0.0",
        "SdkVersionGreater": false,
        "SdkVersionLess": true,
        "CompilationStatusId": 3
      },
      "DeploymentStatus": "Installed",
      "Actions": {
        "CanEdit": false,
        "CanDelete": false
      }
    }
  ],
  "Count": 1
}
```


15.6.3. Получение информации о табличном списке

Запрос для получения информации о табличном списке.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:8091/api-studio/siem/tabular-lists/<Идентификатор списка>
```

URL запроса должен содержать идентификатор табличного списка (UUID).

Параметры строки запроса описаны в таблице ниже.

Таблица 122. Параметры строки запроса /api-studio/siem/tabular-lists/<Идентификатор списка>

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 123. Поля ответа на запрос /api-studio/siem/tabular-lists/<Идентификатор списка>

Поле	Тип данных	Описание
`Actions`	Array	Разрешения на изменение табличного списка
`Actions` → `CanDelete`	Bool	—
`Actions` → `CanEdit`	Bool	—
`Actions` → `CanEditContent`	Bool	—
`AssetGroups`	Array	—
`AssetGroups` → `AssetGroups`	Array	—
`AssetGroups` → `HasError`	Bool	—
`CompilationStatus`	Array	Статус валидации (см. ниже)
`CopyOf`	Array	—
`DeploymentStatus`	Enum (String)	Статус установки в MaxPatrol SIEM: — `NotInstalled` — не установлен; — `Installed` — установлен; — `Outdated` — устарел
`Description`	String	Описание
`Fields`	Array	Поля колонок (см. ниже)
`FillType`	Enum	Назначение: — `registry` — справочник; — `correlationRule` — для правил корреляции; — `enrichmentRule` — для правил обогащения; — `assetGrid` — для данных об активах; — `cybsiGrid` — репутационный табличный список

Поле	Тип данных	Описание
Filters	Array	—
Groups	Array	—
Id	UUID	Идентификатор
IncludeNested	Bool	—
ItemType	String	—
MaxSize	Number (int32)	—
MinScore	Number (int32)	—
ObjectId	String	—
Origin	Array	Информация о поставщике
Origin → Name	String	Название поставщика
Origin → Type	String	—
OriginId	String	—
PdqlQuery	String	—
SystemName	String	Название
TotalRows	Number (int32)	—
Ttl	Number (int32)	—
TtlEnabled	Bool	—
TypicalSize	Number (int32)	—
UserCanEditContent	Bool	—
Статус валидации (CompilationStatus)		
CompilationStatusId	Number (int32)	—
SdkVersion	String	—
SdkVersionGreater	Bool	—
SdkVersionLess	Boolean	—

Поле	Тип данных	Описание
SiemProductVersion	String	—
TaxonomyVersion	String	—
Поля колонок (Fields)		
Id	UUID	Идентификатор
IsIndex	Bool	Является ли индексируемой
IsNullable	Bool	Может ли содержать null
IsPrimaryKey	Bool	Является ли ключевой
Mapping	String	—
Name	String	Название
TypeId	Number (int32)	Тип данных

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
GET https://localhost:8091/api-studio/siem/tabular-lists/3fd59caf-80eb-4963-b0ca-4ab87b19b988?contentDatabase=siem_content_for_test
```

Ответ:


```
{
  "Id": "3fd59caf-80eb-4963-b0ca-4ab87b19b988",
  "ObjectId": "LOC-TL-9",
  "OriginId": "d75c0806-ab0c-45e3-aff0-fd2e723ed878",
  "Folder": null,
  "Origin": {
    "Type": "user",
    "Name": "Локальная система"
  },
  "CopyOf": null,
  "SystemName": "TableLiat_Example",
  "Description": null,
  "MaxSize": null,
  "TypicalSize": null,
  "Ttl": null,
```

```
"TtlEnabled": null,
"CompilationStatus": {
  "TaxonomyVersion": null,
  "SdkVersion": null,
  "SiemProductVersion": "24.0.0",
  "SdkVersionGreater": false,
  "SdkVersionLess": true,
  "CompilationStatusId": 3
},
"TotalRows": 3,
"Fields": [
  {
    "Id": "9d2dd4ce-3c0e-4da4-ba0c-32e483a643e5",
    "Name": "First_Column",
    "Mapping": null,
    "TypeId": 1,
    "IsPrimaryKey": true,
    "IsIndex": true,
    "IsNullable": false
  },
  {
    "Id": "dea0273e-0e1d-4478-89b6-c0bdf4c00606",
    "Name": "Second_Column",
    "Mapping": null,
    "TypeId": 2,
    "IsPrimaryKey": false,
    "IsIndex": false,
    "IsNullable": true
  }
],
"Filters": [
  {
    "Id": "First_Column",
    "Name": "First_Column",
    "Type": "String",
    "MultiSelect": false,
    "Placeholder": ""
  },
  {
    "Id": "Second_Column",
    "Name": "Second_Column",
    "Type": "String",
    "MultiSelect": false,
    "Placeholder": "1-5, 8, 11-13"
  },
  {
```

```
      "Id": "ContentType",
      "Name": "Источник",
      "Type": "Select",
      "MultiSelect": true,
      "Placeholder": ""
    },
    {
      "Id": "Activity",
      "Name": "Активность",
      "Type": "Select",
      "MultiSelect": true,
      "Placeholder": ""
    }
  ],
  "Actions": {
    "CanEdit": true,
    "CanEditContent": true,
    "CanDelete": true
  },
  "UserCanEditContent": false,
  "ItemType": null,
  "MinScore": null,
  "FillType": "Registry",
  "PdqlQuery": null,
  "Groups": [],
  "AssetGroups": null,
  "IncludeNested": null,
  "DeploymentStatus": "NotInstalled"
}
```


15.6.4. Экспорт табличного списка

Запрос для экспорта записей из табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/tabular-lists/<Идентификатор списка>/
export
```

URL запроса должен содержать идентификатор табличного списка.

Параметры строки запроса описаны в таблице ниже.

Таблица 124. Параметры строки запроса /api-studio/siem/tabular-lists/<Идентификатор списка>/export

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 125. Параметры в теле запроса /api-studio/siem/tabular-lists/<Идентификатор списка>/export

Параметр	Обязательный	Тип данных	Описание
`sort`	Нет	Array	Сортировка записей
`sort` → `Name`	Нет	String	Название колонки
`sort` → `Type`	Нет	Number (int32)	Тип сортировки в колонке: `0` — по убыванию значений, `1` — по возрастанию
`sort` → `Order`	Нет	Number (int32)	Порядковый номер при сортировке по нескольким колонкам
`filters`	Нет	Array	Фильтр записей
`filters` → `<Название колонки>`	Нет	Array [String]	Значения в указанной колонке
`rowIds`	Нет	Array [UUID]	Идентификаторы записей

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Тело ответа содержит CSV-файл с записями табличного списка.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/tabular-lists/e157f9c4-
a67e-49ca-96d3-943fcbcfb026/export?contentDatabase=siem_content_for_test
{
  "sort": null,
  "filters": {},
```

```
  "rowIds": []
}
```

Ответ:

```
"host";"user";"action";"exe_regex";"proctitle_regex"
"*";"*";"xattr change";".+";"^/var/tmp/mkinitramfs$"
"*";"*";"xattr change";".+";"^\(d-logind\)$"
```

15.6.5. Открытие сессии импорта табличного списка

Запрос для открытия сессии импорта записей в табличный список.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/tabular-lists/<Идентификатор списка>/
import
```

URL запроса должен содержать идентификатор табличного списка.

Параметры строки запроса описаны в таблице ниже.

Таблица 126. Параметры строки запроса api-studio/siem/tabular-lists/<Идентификатор списка>/import

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 127. Параметры в теле запроса api-studio/siem/tabular-lists/<Идентификатор списка>/import

Параметр	Обязательный	Тип данных	Описание
`fileSize`	Да	Number (int64)	Размер CSV-файла с записями в байтах

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 128. Поля ответа на запрос api-studio/siem/tabular-lists/<Идентификатор списка>/import

Поле	Тип данных	Описание
SessionId	UUID	Идентификатор сессии

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/tabular-lists/3fd59caf-80eb-4963-
b0ca-4ab87b19b988/import?contentDatabase=siem_content_for_test
{
  "fileSize":88
}
```

Ответ:

```
{
  "SessionId": "d0c2f332-1903-4471-a1cf-5ebc0c9140cd"
}
```

15.6.6. Загрузка CSV-файла

Запрос для загрузки CSV-файла с записями для импорта в табличный список.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/tabular-lists/<Идентификатор списка>/
import/<Идентификатор сессии>/upload
```

URL запроса должен содержать идентификатор табличного списка и идентификатор сессии импорта записей.

Параметры строки запроса описаны в таблице ниже.

Таблица 129. Параметры строки запроса api-studio/siem/tabular-lists/<Идентификатор списка>/import/<Идентификатор сессии>/upload

Параметр	Обязательный	Тип данных	Описание
contentDatabase	Да	String	Название БД

Параметр	Обязательный	Тип данных	Описание
`isLastPart`	Да	Bool	—

В теле запроса (в представлении `row text`) необходимо указать записи для импорта. Первая строка должна содержать названия колонок, вторая и последующие строки — значения колонок. Одна строка должна соответствовать одной записи. Названия и значения колонок должны быть заключены в кавычки и разделены точкой с запятой.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/tabular-lists/3fd59caf-80eb-4963-
b0ca-4ab87b19b988/import/d0c2f332-1903-4471-a1cf-5ebc0c9140cd/upload?
contentDatabase=siem_content_for_test
"First_column";"Second_column"
"Row 1";"212"
"Row 2";"85"
"Row 3";"06"
```

15.6.7. Запуск импорта табличного списка

Запрос для запуска импорта записей в табличный список.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/tabular-lists/<Идентификатор списка>/
import/<Идентификатор сессии>
```

URL запроса должен содержать идентификатор табличного списка и идентификатор сессии импорта записей.

Параметры строки запроса описаны в таблице ниже.

Таблица 130. Параметры строки запроса api-studio/siem/tabular-lists/<Идентификатор списка>/import/<Идентификатор сессии>

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 131. Параметры в теле запроса api-studio/siem/tabular-lists/<Идентификатор списка>/import/<Идентификатор сессии>

Параметр	Обязательный	Тип данных	Описание
`matchingRowsActivation`	Да	Enum (String)	Выбор действия с записями при совпадении значений ключевых колонок (для табличного списка типа «справочник»): — `Activate` — оставить обе записи и активировать; — `Deactivate` — оставить обе записи и деактивировать; — `LeaveAsIs` — перезаписать существующую запись

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 132. Поля ответа на запрос api-studio/siem/tabular-lists/<Идентификатор списка>/import/<Идентификатор сессии>

Поле	Тип данных	Описание
`Total`	Number (int64)	Количество записей в файле
`Successful`	Number (int64)	Количество импортированных записей
`Failed`	Number (int64)	Количество пропущенных записей

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/tabular-lists/3fd59caf-80eb-4963-
b0ca-4ab87b19b988/import/d0c2f332-1903-4471-a1cf-5ebc0c9140cd?
contentDatabase=siem_content_for_test
{
  "matchingRowsActivation": "LeaveAsIs"
}
```

Ответ:

```
{
  "Total": 3,
  "Successful": 3,
  "Failed": 0
}
```

15.6.8. Установка объектов в MaxPatrol SIEM

Запрос для установки объектов БД в MaxPatrol SIEM.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/deploy
```

Параметры строки запроса описаны в таблице ниже.

Таблица 133. Параметры строки запроса /api-studio/siem/deploy

Параметр	Обязательный	Тип данных	Описание
contentDatabase	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 134. Параметры в теле запроса /api-studio/siem/deploy

Параметр	Обязательный	Тип данных	Описание
`mode`	Да	Enum (String)	Выбор режима установки: — `all` — все объекты БД; — `group` — все объекты из указанной папки; — `selection` — указанные объекты; — `uninstall` — удаление объектов из MaxPatrol SIEM; — `sync` — синхронизация статусов
`include`	Нет	Array [UUID]	Идентификаторы объектов
`groupId`	Нет	UUID	Идентификатор группы
`filter`	Нет	Array	Фильтр объектов

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 135. Поля ответа на запрос /api-studio/siem/deploy

Поле	Тип данных	Описание
`Id`	UUID	Идентификатор установки

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/deploy?
contentDatabase=siem_content_for_test
{
  "mode": "selection",
  "include": [
    "a6ed8e49-9e25-486b-84ac-2e86c9778b1c"
```

```
    ]
  }
```

Ответ:

```
  {
    "Id": "86aa8d46-4431-4ec8-a2a4-79e6e49b2715"
  }
```

15.6.9. Проверка статуса установки объектов в MaxPatrol SIEM

Запрос для проверки статуса установки объектов БД в MaxPatrol SIEM.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>:8091/api-studio/siem/deploy/statuses
```

Параметры строки запроса описаны в таблице ниже.

Таблица 136. Параметры строки запроса /api-studio/siem/deploy/statuses

Параметр	Обязательный	Тип данных	Описание
`contentDatabase`	Да	String	Название БД

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 137. Параметры в теле запроса /api-studio/siem/deploy/statuses

Параметр	Обязательный	Тип данных	Описание
`<Идентификатор объекта>`	Да	UUID	Идентификатор объекта БД

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 138. Поля ответа на запрос /api-studio/siem/deploy/statuses

Поле	Тип данных	Описание
`<Идентификатор объекта>`	Enum (String)	Статус установки объекта БД: — `Installed` — установлен; — `NotInstalled` — не установлен

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost:8091/api-studio/siem/deploy/statuses?
contentDatabase=siem_content_for_test
[
  "a6ed8e49-9e25-486b-84ac-2e86c9778b1c",
  "cfa4504c-a157-412e-9e66-8eeea39d16d9",
  "be8d2050-a594-4076-9012-279349cff314",
  "201a84b3-f558-452e-b01a-b66e63b5fa0d",
  "dd6f7651-f54d-4303-a8ea-3d554b80dfed"
]
```

Ответ:

```
{
  "201a84b3-f558-452e-b01a-b66e63b5fa0d": "Installed",
  "a6ed8e49-9e25-486b-84ac-2e86c9778b1c": "Installed",
  "be8d2050-a594-4076-9012-279349cff314": "NotInstalled",
  "cfa4504c-a157-412e-9e66-8eeea39d16d9": "Installed",
  "dd6f7651-f54d-4303-a8ea-3d554b80dfed": "NotInstalled"
}
```

15.7. Работа с инцидентами

С помощью запросов к API вы можете регистрировать инциденты, изменять и удалять данные об инцидентах, получать данные о зарегистрированных ранее инцидентах. Кроме того, вы можете экспортировать данные об инцидентах в файл или импортировать их из файла.

Для импорта инцидентов из файла необходимо:

1. Открыть сессию импорта инцидентов.
2. Объявить загрузку JSON-файла с инцидентами.

3. Загрузить файл с инцидентами.
4. Запустить импорт инцидентов.

В этом разделе

15.7.1. Регистрация инцидента

Запрос для регистрации инцидента.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/incidents
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 139. Параметры в теле запроса /api/incidents

Параметр	Обязательный	Тип данных	Описание
`assigned`	Нет	UUID	Ответственный за расследование инцидента
`attackers`	Да	Array	Атакующие активы (см. ниже)
`description`	Нет	String	Текстовый комментарий
`detected`	Да	DateTime	Дата и время регистрации
`groups`	Нет	Array [UUID]	Идентификаторы групп, в которые будет добавлен инцидент
`influence`	Нет	Enum (String)	Влияние инцидента: — `null` — инцидент не имеет последствий; — `Influence1` — потенциальное воздействие на не критически важный актив; — `Influence2` — воздействие на не критически важный актив с последующим нарушением целостности; — `Influence3` — воздействие на не критически важный актив с последующим нарушением конфиденциальности; — `Influence4` — воздействие на критически важный актив, приводящее к финансовым или иным потерям
`name`	Да	String	Название инцидента
`parameters`	Нет	Array	Дополнительные поля (см. ниже)
`severity`	Да	Enum	Уровень опасности инцидента: — `info` — информация; — `low` — низкая; — `medium` — средняя; — `high` — высокая. **Примечание.** Вы можете получить перечень уровней опасности по запросу `/api/incident_dictionaries/severities`

Параметр	Обязательный	Тип данных	Описание
source	Да	Enum (String)	Источник данных об инциденте: — Import — импорт; — NetFor — PT NAD; — SiemScript — MaxPatrol SIEM; — Unknown — неизвестен; — User — оператор
targets	Да	Array	Атакованные активы (см. ниже)
type	Да	Enum	Тип инцидента (см. приложение Е). **Примечание.** Вы можете получить перечень типов по запросу api/incident_dictionaries/types
Атакующие активы (attackers)			
addresses	Нет	Array [String]	IP-адреса
assets	Нет	Array [UUID]	Идентификаторы активов
groups	Нет	Array [UUID]	Идентификаторы групп активов
networks	Нет	Array [UUID]	Идентификаторы сетей
others	Нет	Array [String]	Прочие активы и сети
Дополнительные поля инцидента (parameters)			
id	Да	String	Идентификатор поля
value	Нет	String	Значение поля
Связанные события (events)			
id	Да	UUID	Идентификатор события
timestamp	Нет	DateTime	Дата и время регистрации события
Атакованные активы (targets)			
addresses	Нет	Array [String]	IP-адреса
assets	Нет	Array [UUID]	Идентификаторы активов

Параметр	Обязательный	Тип данных	Описание
groups	Нет	Array [UUID]	Идентификаторы групп активов
networks	Нет	Array [UUID]	Идентификаторы сетей
others	Нет	Array [String]	Прочие активы и сети

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 140. Поля ответа на запрос /api/incidents

Поле	Тип данных	Описание
id	UUID	Идентификатор инцидента
key	String	Ключ инцидента

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Пример

Запрос:

```
POST https://localhost/api/incidents
{
  "description": "Это описание",
  "source": "user",
  "name": "Название",
  "detected": "2021-05-12T21:00:34.000Z",
  "type": "PatchPolicyViolation",
  "influence": "Influence1",
  "severity": "Medium",
  "groups": ["1499141b-4c00-0001-0000-00000000000d"],
  "parameters": [],
  "assigned": "2044561a-fd1e-434a-b83a-ac37b820720d",
  "targets": {
    "groups": ["149f37fa-c2c0-0001-0000-000000000011"],
    "assets": ["1493f168-02c0-0001-0000-00000000014c"],
    "networks": ["1493f393-0980-0001-0000-00000000034e"],
```

```
      "addresses": ["1.1.1.1"],
      "others": ["2.2.2.2"]
    },
    "attackers": {
      "groups": ["14952148-f980-0001-0000-000000000009"],
      "assets": ["1493f168-0940-0001-0000-00000000014f"],
      "networks": ["1493f393-0bc0-0001-0000-00000000034f"],
      "addresses": ["3.3.3.3"],
      "others": ["4.4.4.4"]
    }
  }
```


Ответ:


```
  {
    "id": "cc9e3cbd-06be-48f1-8092-363afb95ee19",
    "key": "INC-5"
  }
```


См. также

Правила заполнения полей инцидента category и type (см. приложение E)

15.7.2. Изменение инцидента

Запрос для изменения инцидента.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/incidents/<Идентификатор инцидента>
```

URL запроса должен содержать идентификатор инцидента (UUID).

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 141. Параметры в теле запроса /api/incidents/<Идентификатор инцидента>

Параметр	Обязательный	Тип данных	Описание
assigned	Нет	UUID	Ответственный за расследование инцидента
attackers	Да	Array	Атакующие активы (см. ниже)
description	Нет	String	Текстовый комментарий
detected	Да	DateTime	Дата и время регистрации
groups	Нет	Array [UUID]	Идентификаторы групп, в которые будет добавлен инцидент

Параметр	Обязательный	Тип данных	Описание
`influence`	Нет	Enum (String)	Влияние инцидента: — `null` — инцидент не имеет последствий; — `Influence1` — потенциальное воздействие на не критически важный актив; — `Influence2` — воздействие на не критически важный актив с последующим нарушением целостности; — `Influence3` — воздействие на не критически важный актив с последующим нарушением конфиденциальности; — `Influence4` — воздействие на критически важный актив, приводящее к финансовым или иным потерям
`name`	Да	String	Название инцидента
`parameters`	Нет	Array	Дополнительные поля (см. ниже)
`severity`	Да	Enum	Уровень опасности инцидента: — `info` — информация; — `low` — низкая; — `medium` — средняя; — `high` — высокая. **Примечание.** Вы можете получить перечень уровней опасности по запросу `/api/incident_dictionaries/severities`
`source`	Да	Enum (String)	Источник данных об инциденте: — `Import` — импорт; — `NetFor` — PT NAD; — `SiemScript` — MaxPatrol SIEM; — `Unknown` — неизвестен; — `User` — оператор
`targets`	Да	Array	Атакованные активы (см. ниже)
`type`	Да	Enum	Тип инцидента (см. приложение Е).

Параметр	Обязательный	Тип данных	Описание
			Примечание. Вы можете получить перечень типов по запросу `api/incident_dictionaries/types`
Атакующие активы (attackers)			
`addresses`	Нет	Array [String]	IP-адреса
`assets`	Нет	Array [UUID]	Идентификаторы активов
`groups`	Нет	Array [UUID]	Идентификаторы групп активов
`networks`	Нет	Array [UUID]	Идентификаторы сетей
`others`	Нет	Array [String]	Прочие активы и сети
Дополнительные поля инцидента (parameters)			
`id`	Да	String	Идентификатор поля
`value`	Нет	String	Значение поля
Атакованные активы (targets)			
`addresses`	Нет	Array [String]	IP-адреса
`assets`	Нет	Array [UUID]	Идентификаторы активов
`groups`	Нет	Array [UUID]	Идентификаторы групп активов
`networks`	Нет	Array [UUID]	Идентификаторы сетей
`others`	Нет	Array [String]	Прочие активы и сети

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Пример

Запрос:

```
PUT https://localhost/api/incidents/cc9e3cbd-06be-48f1-8092-363afb95ee19
{
  "name": "Название Simple444",
  "source": "user",
  "detected": "2021-07-12T21:00:34.000Z",
  "type": "PatchPolicyViolation",
  "severity": "Low",
  "influence":"Influence2",
  "targets": {
    "groups": [],
    "assets": [],
    "networks": [],
    "addresses": ["10.0.11.112"],
    "others": ["10.2.12.44"]
  },
  "attackers": {
    "groups": [],
    "assets": [],
    "networks": [],
    "addresses": ["10.5.13.108"],
    "others": ["10.0.14.1"]
  }
}
```

15.7.3. Удаление инцидентов

Запрос для удаления инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v2/incidents/delete_by_ids
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 142. Параметры в теле запроса /api/v2/incidents/delete_by_ids

Параметр	Обязательный	Тип данных	Описание
incidents	Да	Array [UUID]	Идентификаторы инцидентов

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Пример

Запрос:

```
POST https://localhost/api/v2/incidents/delete_by_ids
{
  "incidents": [
    "a5dc33db-b96e-4117-be0f-6b6b0e2a74e0"
  ]
}
```

15.7.4. Получение списка инцидентов

Запрос для получения списка инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v2/incidents
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 143. Параметры в теле запроса /api/v2/incidents

Параметр	Обязательный	Тип данных	Описание
`filter`	Да	Array	Фильтр (см. ниже)
`filterTimeType`	Да	Enum (String)	Свойство инцидента, попадающее во временной интервал: — `Approval` — дата утверждения; — `Closing` — дата закрытия; — `Creation` — дата регистрации; — `Detection` — дата обнаружения; — `InProgress` — дата взятия в работу;

Параметр	Обязательный	Тип данных	Описание
			— `LastModification` — дата последнего изменения; — `Resolving` — дата разрешения
`groups`	Нет	Array	Группы инцидентов (или активов, с которыми связаны инциденты)
`limit`	Нет	Number	Максимальное количество инцидентов в ответе
`offset`	Нет	Number	Количество отбрасываемых инцидентов
`queryIds`	Нет	Array	Идентификаторы фильтров инцидентов. **Примечание.** Список идентификаторов фильтров вы можете получить по GET-запросу `api/v1/incidents/filters_hierarchy`
`timeFrom`	Да	DateTime	Начало временного интервала (например `2020-12-31T21:00:00.000Z`)
`timeTo`	Нет	DateTime	Конец временного интервала
Фильтр инцидентов (filter)			
`orderby`	Нет	Array	Сортировка в списке
`orderby` → `field`	Нет	String	Поле, по которому нужно сортировать
`orderby` → `sortOrder`	Нет	Enum (String)	Порядок сортировки: — `ascending` — по возрастанию; — `descending` — по убыванию
`select`	Да	Array [String]	Список полей инцидента, из которых нужно получить данные
`where`	Нет	String	Условие фильтрации (предикат фильтра)

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 144. Поля ответа на запрос /api/v2/incidents

Поле	Тип данных	Описание
`totalItems`	Number	Количество инцидентов, удовлетворяющих указанным условиям фильтрации

Поле	Тип данных	Описание
`incidents`	Array	Список инцидентов (см. ниже)
Данные инцидента (incidents)		
`assigned`	Array	Данные ответственного за расследование инцидента (см. ниже)
`category`	Enum	Категория инцидента
`created`	DateTime	Дата и время регистрации инцидента
`id`	UUID	Идентификатор инцидента
`isConfirmed`	Bool	Подтвержден ли инцидент
`key`	String	Идентификатор инцидента, например `INC-24453`
`name`	String	Название инцидента
`severity`	Enum	Уровень опасности инцидента: — `low` — низкая; — `medium` — средняя; — `high` — высокая. **Примечание.** Вы можете получить перечень уровней опасности по запросу `/api/incident_dictionaries/severities`
`status`	Enum (String)	Статус инцидента: — `New` — новый; — `Approved` — утвержден; — `InProgress` — в работе; — `Resolved` — разрешен; — `Closed` — закрыт
`type`	Enum	Тип инцидента (см. приложение Е). **Примечание.** Вы можете получить перечень типов по запросу `api/incident_dictionaries/types`
Данные ответственного за расследование (incidents → assigned)		
`email`	String	Адрес электронной почты
`firstName`	String	Имя
`id`	UUID	Идентификатор
`lastName`	String	Фамилия

Поле	Тип данных	Описание
phone	String	Телефон

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v2/incidents
{
  "offset": 0,
  "limit": 1,
  "groups": {
    "filterType": "no_filter"
  },
  "timeFrom": "2020-12-31T21:00:00.000Z",
  "timeTo": null,
  "filterTimeType": "creation",
  "filter": {
    "select": [
      "key",
      "name",
      "category",
      "type",
      "status",
      "created",
      "assigned"
    ],
    "where": "",
    "orderby": [
      {
        "field": "created",
        "sortOrder": "descending"
      },
      {
        "field": "status",
        "sortOrder": "ascending"
      },
      {
        "field": "severity",
        "sortOrder": "descending"
```

```
        }
      ]
    },
    "queryIds": [
      "all_incidents"
    ]
  }
```


Ответ:


```
  {
    "incidents": [
      {
        "id": "678123d6-c767-4983-89c2-2282e7e46b5b",
        "key": "INC-3",
        "name": "Пример",
        "isConfirmed": false,
        "status": "New",
        "category": "PolicyViolation",
        "type": "PatchPolicyViolation",
        "created": "2021-05-13T13:37:32.0730000Z",
        "assigned": {
          "id": null,
          "firstName": null,
          "lastName": null,
          "email": null,
          "phone": null
        },
        "severity": "High"
      }
    ],
    "totalItems": 3
  }
```


15.7.5. Получение данных инцидента

Запрос для получения данных инцидента.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/incidentsReadModel/incidents/<Идентификатор инцидента>
```

URL запроса должен содержать идентификатор инцидента (UUID).

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 145. Поля ответа на запрос /api/incidentsReadModel/incidents/<Идентификатор инцидента>

Поле	Тип данных	Описание
`assigned`	Array	Данные ответственного за расследование инцидента
`attackers`	Array	Атакующие активы (см. ниже)
`category`	Enum (String)	Категория инцидента
`correlationRuleNames`	Array [String]	Правила корреляции, по которым зарегистрирован инцидент
`created`	DateTime	Дата и время регистрации инцидента
`description`	String	Текстовый комментарий
`detected`	DateTime	Дата и время регистрации
`groups`	Array	Идентификаторы групп активов
`id`	UUID	Идентификатор инцидента
`influence`	String	Влияние инцидента: — `null` — инцидент не имеет последствий; — `Influence1` — потенциальное воздействие на не критически важный актив; — `Influence2` — воздействие на не критически важный актив с последующим нарушением целостности; — `Influence3` — воздействие на не критически важный актив с последующим нарушением конфиденциальности; — `Influence4` — воздействие на критически важный актив, приводящее к финансовым или иным потерям
`isConfirmed`	Bool	Подтвержден ли инцидент
`key`	String	Идентификатор инцидента, например `INC-24453`
`measures`	String	Меры, принятые для решения инцидента
`modified`	Array	История изменений данных инцидента
`name`	String	Название инцидента
`parameters`	Array	Дополнительные поля (см. ниже)

Поле	Тип данных	Описание
`reporter`	Array	Пользователь, зарегистрировавший инцидент
`severity`	String	Опасность: — `low` — низкая; — `medium` — средняя; — `high` — высокая. **Примечание.** Вы можете получить перечень уровней опасности по запросу `/api/incident_dictionaries/severities`
`source`	String	Источник данных об инциденте: — `Import` — импорт; — `NetFor` — PT NAD; — `SiemScript` — MaxPatrol SIEM; — `Unknown` — неизвестен; — `User` — оператор
`status`	Enum (String)	Статус инцидента: — `New` — новый; — `Approved` — утвержден; — `InProgress` — в работе; — `Resolved` — разрешен; — `Closed` — закрыт
`targets`	Array	Атакованные активы (см. ниже)
`type`	String	Тип инцидента (см. приложение Е). **Примечание.** Вы можете получить перечень типов по запросу `api/incident_dictionaries/types`
`unitedNotificationCount`	Number (int32)	Количество объединенных нотификаций. Для автоинцидентов значение параметра отлично от `null`, для остальных — `null`
Атакующие активы (attackers)		
`addresses`	Array [String]	IP-адреса
`assets`	Array [UUID]	Идентификаторы активов

Поле	Тип данных	Описание
groups	Array [UUID]	Идентификаторы групп активов
networks	Array [UUID]	Идентификаторы сетей
others	Array [String]	Прочие активы и сети
Дополнительные поля инцидента (parameters)		
id	String	Идентификатор поля
value	String	Значение поля
Атакованные активы (targets)		
addresses	Array [String]	IP-адреса
assets	Array [UUID]	Идентификаторы активов
groups	Array [UUID]	Идентификаторы групп активов
networks	Array [UUID]	Идентификаторы сетей
others	Array [String]	Прочие активы и сети

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/incidentsReadModel/incidents/
917490b0-1a4e-47f8-82d8-7d2031a1c518
```

Ответ:

```
{
  "id": "917490b0-1a4e-47f8-82d8-7d2031a1c518",
  "key": "INC-22",
  "name": "Пример инцидента",
  "source": "import",
  "description": "",
```

```
"isConfirmed": false,
"status": "New",
"measures": "",
"category": "Attack",
"severity": "Medium",
"type": "Spam",
"created": "2021-03-23T08:43:11.0430000Z",
"detected": "2021-03-15T21:43:00.0000000Z",
"modified": {
  "date": "2021-05-13T14:31:04.1367790Z",
  "user": null,
  "properties": [
    {
      "name": "Name",
      "oldValue": "Test_inn22",
      "newValue": "Test_inn33"
    }
  ]
},
"reporter": null,
"assigned": null,
"influence": null,
"groups": [
  {
    "type": "GroupInfo",
    "groupType": "static",
    "id": "1499141b-4c00-0001-0000-00000000000d",
    "name": "Import",
    "accessibility": "granted"
  }
],
"targets": {
  "groups": [],
  "assets": [],
  "networks": [],
  "addresses": [],
  "others": []
},
"attackers": {
  "groups": [],
  "assets": [],
  "networks": [],
  "addresses": [],
  "others": []
},
"parameters": [
```

```
      {
        "id": "SpamMailCount",
        "value": ""
      }
    ],
    "unitedNotificationCount": null,
    "correlationRuleNames": []
  }
```


15.7.6. Получение событий по инциденту

Запрос для получения списка событий, связанных с инцидентом.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/incidents/<Идентификатор инцидента>/events
```

URL запроса должен содержать идентификатор инцидента.

Параметры строки запроса описаны в таблице ниже.

Таблица 146. Параметры строки запроса /api/incidents/<Идентификатор инцидента>/events

Параметр	Обязательный	Тип данных	Описание
`limit`	Нет	Number	Максимальное число событий в ответе
`offset`	Нет	Number	Количество отбрасываемых событий

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 147. Поля ответа на запрос /api/incidents/<Идентификатор инцидента>/events

Поле	Тип данных	Описание
`date`	DateTime	Дата и время регистрации события из поля `time`
`description`	String	Текст события из поля `text`
`id`	UUID	Идентификатор события из поля `uuid`

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/incidents/1f6e44d8-4585-4c17-8104-fc2efc3a9216/events?
limit=5
```

Ответ:

```
[
  {
    "id": "00000006-0d99-00e2-f000-00009d368550",
    "date": "2021-06-28T09:05:37.0000000Z",
    "description": "Kerberos-аутентификация пользователя ExampleUserName, запрошенная
с узла 10.0.110.10, прошла успешно. Служба Kerberos на узле example.host.ru выдала
билет проверки подлинности Kerberos (TGT)"
  },
  {
    "id": "00000006-0d99-00c7-f000-00019d368550",
    "date": "2021-06-28T09:05:09.0000000Z",
    "description": "Пользователь ExampleUserName осуществил успешный вход в систему
на узле example.host.ru. Тип входа: Network"
  }
]
```

15.7.7. Экспорт инцидентов

Запрос для экспорта инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v1/incidents/export/export_by_filter
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 148. Параметры в теле запроса /api/v1/incidents/export/export_by_filter

Параметр	Обязательный	Тип данных	Описание
filter	Да	Array	Фильтр (см. ниже)

Параметр	Обязательный	Тип данных	Описание
filterTimeType	Да	Enum (String)	Свойство инцидента, попадающее во временной интервал: — Approval — дата утверждения; — Closing — дата закрытия; — Creation — дата регистрации; — Detection — дата обнаружения; — InProgress — дата взятия в работу; — LastModification — дата последнего изменения; — Resolving — дата разрешения
groups	Нет	Array	Группы инцидентов (или активов, с которыми связаны инциденты)
limit	Нет	Number	Максимальное количество инцидентов в ответе
offset	Нет	Number	Количество отбрасываемых инцидентов
queryIds	Нет	Array	Идентификаторы фильтров инцидентов. **Примечание.** Список идентификаторов фильтров вы можете получить по GET-запросу api/v1/incidents/filters_hierarchy
timeFrom	Да	DateTime	Начало временного интервала (например 2020-12-31T21:00:00.000Z)
timeTo	Нет	DateTime	Конец временного интервала
Фильтр инцидентов (filter)			
orderby	Нет	Array	Сортировка в списке
orderby → field	Нет	String	Поле, по которому нужно сортировать
orderby → sortOrder	Нет	Enum (String)	Порядок сортировки: — ascending — по возрастанию; — descending — по убыванию
select	Да	Array [String]	Список полей инцидента, из которых нужно получить данные
where	Нет	String	Условие фильтрации (предикат фильтра)

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Тело ответа содержит файл формата JSON с данными инцидентов. Файл может содержать поля, описанные в таблице ниже.

Таблица 149. Поля файла Инциденты_<Дата экспорта>.json

Поле	Тип данных	Описание
`formatVersion`	String	Версия формата экспорта
`producer` → `type`	String	Система, из которой импортированы инциденты
`producer` → `id`	UUID	Идентификатор системы
`content`	Array [String]	Тип содержимого
`incidents`	Array	Список инцидентов
`incidents` → `identification`	Array	Данные для идентификации
`incidents` → `identification` → `source`	Array	—
`incidents` → `identification` → `id`	String	Идентификатор
`incidents` → `content`	Array	Данные инцидента (см. ниже)
Данные инцидента (incidents → content)		
`actionsTaken`	String	Приняты меры по расследованию: — `Yes` — да; — `No` — нет; — `Unknown` — неизвестно
`affectedEntities`	Array	Затронутые объекты
`assignedTo`	Array	Данные ответственного за расследование инцидента
`attackingEntities`	Array	—
`confirmed`	Bool	Утвержден
`createdAt`	DateTime	—
`createdBy`	Array	—

Поле	Тип данных	Описание
description	String	Текстовый комментарий
influence	Array	Влияние инцидента: — null — инцидент не имеет последствий; — Influence1 — потенциальное воздействие на не критически важный актив; — Influence2 — воздействие на не критически важный актив с последующим нарушением целостности; — Influence3 — воздействие на не критически важный актив с последующим нарушением конфиденциальности; — Influence4 — воздействие на критически важный актив, приводящее к финансовым или иным потерям
modifiedAt	DateTime	Дата и время последнего изменения
name	String	Название
registeredAt	DateTime	Дата и время регистрации
severity	String	Опасность: — low — низкая; — medium — средняя; — high — высокая
state	Array	Статус инцидента: — New — новый; — Approved — утвержден; — InProgress — в работе; — Resolved — разрешен; — Closed — закрыт
type	Array	Тип инцидента
typeSpecificParams	Array	Дополнительные параметры инцидента

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v1/incidents/export/export_by_filter
{
  "offset": 0,
  "limit": 50,
  "groups": {
    "filterType": "no_filter"
  },
  "timeFrom": "2020-12-31T21:00:00.000Z",
  "timeTo": null,
  "filterTimeType": "creation",
  "filter": {
    "select": [
      "key",
      "name",
      "category",
      "type",
      "status",
      "created",
      "assigned"
    ],
    "where": "",
    "orderby": [
      {
        "field": "created",
        "sortOrder": "descending"
      },
      {
        "field": "status",
        "sortOrder": "ascending"
      },
      {
        "field": "severity",
        "sortOrder": "descending"
      }
    ]
  },
  "queryIds": [
    "all_incidents"
  ]
}
```

Ответ:


```
{
  "formatVersion": "1.0",
  "producer": {
    "type": "PT.MPX",
    "id": "5a0a9d31-4e18-4b50-9ff0-502a2f4a071d"
  },
  "content": ["incidents"],
  "incidents": [ {
      "identification": {
        "source": {
          "type": "PT.MPX",
          "id": "5a0a9d31-4e18-4b50-9ff0-502a2f4a071d"
        },
        "id": "INC-2"
      },
      "content": {
        "name": "Название",
        "description": "",
        "state": {
          "id": "New"
        },
        "actionsTaken": "",
        "confirmed": false,
        "registeredAt": "2021-07-12T21:00:34Z",
        "createdAt": "2021-07-21T12:16:47.081Z",
        "modifiedAt": "2021-07-21T14:07:14.52149Z",
        "influence": {
          "id": "Influence2"
        },
        "severity": "Low",
        "type": {
          "id": "PatchPolicyViolation"
        },
        "typeSpecificParams": [],
        "affectedEntities": {
          "addresses": ["10.0.11.112"],
          "otherAssetsAndNetworks": ["10.2.12.44"]
        },
        "attackingEntities": {
          "addresses": ["10.5.13.108"],
          "otherAssetsAndNetworks": ["10.0.14.1"]
        },
        "createdBy": {
          "id": "bcfac8ac-3c0e-40d5-a13e-56252b4e2dd7"
        },
```

```
        "assignedTo": {
          "id": "bcfac8ac-3c0e-40d5-a13e-56252b4e2dd7"
        }
      }
    }
  ]
}
```

15.7.8. Открытие сессии импорта инцидентов

Запрос для открытия сессии импорта инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v1/incidents/import/sessions
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 150. Поля ответа на запрос /api/v1/incidents/import/sessions

Поле	Тип данных	Описание
`sessionId`	UUID	Идентификатор сессии

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v1/incidents/import/sessions
```

Ответ:

```
{
  "sessionId": "37b3fc33-e7ef-42e6-bd9b-8ee6b62b0c93"
}
```

15.7.9. Объявление загрузки файла с инцидентами

Запрос для объявления загрузки файла с инцидентами.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v1/incidents/import/sessions/<Идентификатор сессии>/files
```

URL запроса должен содержать идентификатор сессии импорта инцидентов.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 151. Параметры в теле запроса api/v1/incidents/import/sessions/<Идентификатор сессии>/files

Параметр	Обязательный	Тип данных	Описание
`fileName`	Да	String	Имя файла

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 152. Поля ответа на запрос /api/v1/incidents/import/sessions/<Идентификатор сессии>/files

Поле	Тип данных	Описание
`fileId`	UUID	Идентификатор файла

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v1/incidents/import/sessions/37b3fc33-e7ef-42e6-
bd9b-8ee6b62b0c93/files
{
  "fileName": "My_Incident.json"
}
```

Ответ:

```
{
  "fileId": "54ca9e65-48b4-44c3-bd1f-2b56acf0fa7b"
}
```

15.7.10. Загрузка файла с инцидентами

Запрос для загрузки файла с инцидентами.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v1/incidents/import/sessions/<Идентификатор сессии>/
uploads
```

URL запроса должен содержать идентификатор сессии импорта инцидентов (UUID).

Параметры строки запроса описаны в таблице ниже.

Таблица 153. Параметры строки запроса /api/v1/incidents/import/sessions/<Идентификатор сессии>/uploads

Параметр	Обязательный	Тип данных	Описание
`chunkNumber`	Да	Number	Номер фрагмента
`fileId`	Да	UUID	Идентификатор файла
`format`	Да	String	Формат данных инцидентов (по умолчанию `unified`)
`totalChunks`	Да	Number	Количество фрагментов

В теле запроса необходимо передать файл с инцидентами в формате JSON и описанные в таблице ниже параметры в представлении `form-data`.

Таблица 154. Параметры в теле запроса /api/v1/incidents/import/sessions/<Идентификатор сессии>/uploads

Параметр	Обязательный	Тип данных	Описание
`flowChunkNumber`	Да	Number	Номер фрагмента
`flowChunkSize`	Да	Number	Максимальный размер фрагмента
`flowCurrentChunkSize`	Да	Number	Размер передаваемого фрагмента
`flowFilename`	Да	String	Имя файла

Параметр	Обязательный	Тип данных	Описание
`flowIdentifier`	Да	String	Идентификатор файла
`flowRelativePath`	Да	String	Относительный путь к файлу
`flowTotalChunks`	Да	Number	Количество фрагментов

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 155. Поля ответа на запрос /api/v1/incidents/import/sessions/<Идентификатор сессии>/uploads

Поле	Тип данных	Описание
`isValid`	Bool	Удачно ли завершилась проверка файла

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v1/incidents/import/sessions/37b3fc33-e7ef-42e6-
bd9b-8ee6b62b0c93/uploads?fileId=54ca9e65-48b4-44c3-
bd1f-2b56acf0fa7b&chunkNumber=1&format=unified&totalChunks=1
flowChunkNumber:1
flowChunkSize:946
flowCurrentChunkSize:946
flowIdentifier:946-My_Incidentjson
flowFilename:My_Incident.json
flowRelativePath:My_Incident.json
flowTotalChunks:1
file:My_Incident.json
```

Ответ:

```
{
  "isValid": true
}
```

15.7.11. Импорт инцидентов

Запрос для импорта инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v1/incidents/import/complete_import
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 156. Параметры в теле запроса api/v1/incidents/import/complete_import

Параметр	Обязательный	Тип данных	Описание
`sessionId`	Да	UUID	Идентификатор сессии
`groups`	Да	Array [UUID]	Идентификаторы групп в которые добавляются инциденты
`mode`	Да	Enum (String)	Действие при совпадении инцидентов: — `Replace` — заменять; — `Skip` — пропускать
`format`	Да	Enum	Формат данных инцидентов (по умолчанию `unified`)

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 157. Поля ответа на запрос /api/v1/incidents/import/complete_import

Поле	Тип данных	Описание
`importResult`	Array	Результат импорта
`importTime`	DateTime	Дата и время начала импорта
`totalImportedIncidentCount`	Number	Количество импортированных инцидентов
`totalIncidentCount`	Number	Количество инцидентов в файле
`totalSkippedIncidentCount`	Number	Количество пропущенных инцидентов

Поле	Тип данных	Описание
Результат импорта (importResult)		
`fileId`	UUID	Идентификатор файла
`importedIncidentCount`	Number	Количество импортированных инцидентов
`skippedIncidentCount`	Number	Количество пропущенных инцидентов
`success`	Bool	Успешно ли выполнен импорт
`totalIncidentCount`	Number	Количество инцидентов в файле

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/v1/incidents/import/complete_import
{
  "sessionId": "37b3fc33-e7ef-42e6-bd9b-8ee6b62b0c93",
  "groups": ["1499141b-4c00-0001-0000-00000000000d"],
  "mode": "replace",
  "format": "unified"
}
```

Ответ:

```
{
  "importTime": "2021-05-18T13:44:45.2865836Z",
  "totalImportedIncidentCount": 1,
  "totalSkippedIncidentCount": 0,
  "totalIncidentCount": 1,
  "importResult": [
    {
      "fileId": "54ca9e65-48b4-44c3-bd1f-2b56acf0fa7b",
      "importedIncidentCount": 1,
      "skippedIncidentCount": 0,
      "totalIncidentCount": 1,
      "success": true
    }
  ]
}
```

15.7.12. Удаление инцидентов по фильтру

Запрос для удаления инцидентов по фильтру.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v2/incidents/delete_by_filter
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 158. Параметры в теле запроса /api/v2/incidents/delete_by_filter

Параметр	Обязательный	Тип данных	Описание
`timeFrom`	Да	String	Начало временного интервала
`timeTo`	Нет	String	Конец временного интервала
`filterTimeType`	Да	Enum (String)	Свойство инцидента, попадающее во временной интервал: — `creation` — дата регистрации; — `detection` — дата обнаружения; — `lastModification` — дата последнего изменения; — `approval` — дата утверждения; — `inProgress` — дата начала работы; — `resolving` — дата окончания работы; — `closing` — дата закрытия инцидента
`filter`	Да	Array	Фильтр: — `select` — список полей инцидента, из которых нужно получить данные; — `where` — условие фильтрации (предикат фильтра); — `orderby` — сортировка в списке: • `field` — поле; • `sortOrder` — порядок сортировки (`ascending` — по возрастанию, `descending` — по убыванию)

Параметр	Обязательный	Тип данных	Описание
`excludeIncidents`	Нет	Array	Инциденты, исключенные из фильтрации: — `items` — элемент массива
`queryIds`	Нет	Array	Идентификатор запроса: — `items` — элемент массива
`groups`	Да	Array	Группы: — `filterType` — тип фильтра

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 159. Поля ответа на запрос /api/v2/incidents/delete_by_filter

Поле	Тип данных	Описание
`message`	String	Сообщение
`modelState`	Array	Состояние модели: — `additionalProp1`; — `additionalProp2`; — `additionalProp3`

15.7.13. Удаление связи события с инцидентом

Запрос для удаления связи события с инцидентом.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/incidents/{incidentId}/events/{eventId}
```

URL запроса содержит path-параметры `incidentId` — идентификатор инцидента и `eventId` — идентификатор события.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 160. Поля ответа на запрос /api/incidents/{incidentId}/events/{eventId}

Поле	Тип данных	Описание
`message`	String	Сообщение
`modelState`	Array	Состояние модели: — `additionalProp1`; — `additionalProp2`; — `additionalProp3`

15.7.14. Удаление связи актива с инцидентом

Запрос для удаления связи актива с инцидентом.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/incidents/{incidentId}/assets/{assetType}/
{elementType}/{id}
```

URL запроса содержит path-параметры `incidentId` — идентификатор инцидента, `assetType` — тип актива, `elementType` — тип элемента и `id` — идентификатор.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 161. Поля ответа на запрос /api/incidents/{incidentId}/assets/{assetType}/{elementType}/{id}

Поле	Тип данных	Описание
`message`	String	Сообщение
`modelState`	Array	Состояние модели: — `additionalProp1`; — `additionalProp2`; — `additionalProp3`

15.7.15. Изменение мер инцидента

Запрос для изменения мер инцидента.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/incidents/{incidentId}/measures
```

URL запроса содержит path-параметр `incidentId` — идентификатор инцидента.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 162. Параметры в теле запроса /api/incidents/{incidentId}/measures

Параметр	Обязательный	Тип данных	Описание
`measures`	Нет	String	Меры инцидента

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 163. Поля ответа на запрос /api/incidents/{incidentId}/measures

Поле	Тип данных	Описание
`message`	String	Сообщение

Поле	Тип данных	Описание
modelState	Array	Состояние модели: — additionalProp1; — additionalProp2; — additionalProp3

15.7.16. Обновление групп инцидентов

Запрос для обновления групп инцидентов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/incidents/{incidentId}/groups
```

URL запроса содержит path-параметр incidentId — идентификатор инцидента.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 164. Параметры в теле запроса /api/incidents/{incidentId}/groups

Параметр	Обязательный	Тип данных	Описание
groupIds	Да	Array[String]	Идентификаторы групп

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

15.7.17. Привязка событий к инциденту

Запрос для привязки событий к инциденту.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/incidents/events
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 165. Параметры в теле запроса /api/incidents/events

Параметр	Обяза-тельный	Тип данных	Описание
`linkInfo`	Да	Object	Информация о привязке события к инциденту: — `incidentKey` — ключ инцидента; — `events` — список событий: • `id` — системный идентификатор события; • `timestamp` — дата и время привязки к инциденту

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 166. Поля ответа на запрос /api/incidents/events

Поле	Тип данных	Описание
`message`	String	Сообщение
`modelState`	Array	Состояние модели: — `additionalProp1`; — `additionalProp2`; — `additionalProp3`

15.7.18. Изменение статуса инцидента

Запрос для изменения статуса инцидента.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/incidents/{incidentId}/transitions
```

URL запроса содержит path-параметр `incidentId` — идентификатор инцидента.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 167. Параметры в теле запроса /api/incidents/{incidentId}/transitions

Параметр	Обязательный	Тип данных	Описание
id	Нет	String	Идентификатор инцидента
message	Нет	String	Сообщение
measures	Нет	String	Меры

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

400 (Bad request) — синтаксическая ошибка в запросе.

15.7.19. Создание задачи

Запрос для создания задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/issues
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 168. Параметры в теле запроса /api/issues

Параметр	Обязательный	Тип данных	Описание
name	Нет	String	Название инцидента
type	Нет	String	Тип инцидента
estimatedTime	Нет	String	Расчетное время
assigned	Нет	String	Данные ответственного за расследование инцидента
description	Нет	String	Текстовый комментарий
incidentId	Да	String	Идентификатор инцидента

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 169. Поля ответа на запрос /api/issues

Поле	Тип данных	Описание
`id`	Да	Идентификатор задачи
`status`	Нет	Статус задачи
`name`	Нет	Имя задачи
`assigned`	Нет	Данные ответственного за расследование инцидента: — `id` — идентификатор; — `firstName` — имя; — `lastName` — фамилия; — `email` — адрес электронной почты; — `phone` — телефон
`estimatedTime`	Нет	Расчетное время

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.7.20. Обновление задачи

Запрос для обновления задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/issues/{issueId}
```

URL запроса содержит path-параметр `issueId` — идентификатор задачи.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 170. Параметры в теле запроса api/issues/{issueId}

Параметр	Обязательный	Тип данных	Описание
`name`	Нет	String	Название инцидента

Параметр	Обязательный	Тип данных	Описание
type	Нет	String	Тип инцидента
estimatedTime	Нет	String	Расчетное время
assigned	Нет	String	Данные ответственного за расследование инцидента
description	Нет	String	Текстовый комментарий
incidentId	Да	String	Идентификатор инцидента
status	Нет	String	Статус инцидента

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.7.21. Получение данных о событии

Запрос для получения данных о событии.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/incidentsReadModel/v2/incidents/{incidentId}/events
```

URL запроса содержит path-параметр `incidentId` — идентификатор инцидента.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 171. Поля ответа на запрос /api/incidentsReadModel/v2/incidents/{incidentId}/events

Поле	Тип данных	Описание
id	String	Идентификатор события
timestamp	String	Дата и время события

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

Ответ может содержать поля, описанные в таблице ниже.

Таблица 172. Поля ответа на запрос /api/incidentsReadModel/v2/incidents/{incidentId}/events

Поле	Тип данных	Описание
`message`	String	Сообщение
`modelState`	Array	Состояние модели: — `additionalProp1`; — `additionalProp2`; — `additionalProp3`

15.8. Работа с событиями

С помощью запросов к API вы можете получить список событий по фильтру или привязать события к зарегистрированному ранее инциденту.

В этом разделе

Получение списка событий (см. раздел 15.8.1)

Привязка событий к инциденту (см. раздел 15.8.2)

15.8.1. Получение списка событий

Запрос для получения списка событий.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/events/v2/events
```

Параметры строки запроса описаны в таблице ниже.

Таблица 173. Параметры строки запроса api/events/v2/events

Параметр	Обязательный	Тип данных	Описание
`groupIds`	Нет	Array [UUID]	Идентификаторы групп активов

Параметр	Обязательный	Тип данных	Описание
incidentId	Нет	UUID	Идентификатор инцидента, с которым связаны события
limit	Нет	Number	Максимальное количество событий в списке
offset	Нет	Number (int32)	Интервал времени, за который зарегистрированы события
recursive	Нет	Bool	Включать ли вложенные группы активов
token	Нет	UUID	Токен представления

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 174. Параметры в теле запроса /api/events/v2/events

Параметр	Обязательный	Тип данных	Описание
checkpointFrom	Нет	Number (int64)	—
checkpointTo	Нет	Number (int64)	—
filter	Да	Array	Фильтр событий (см. ниже)
groupValues	Нет	Array [UUID]	—
timeFrom	Да	Number (int64)	Начало временного интервала, в который было зарегистрировано событие (Unix-время)
timeTo	Нет	Number (int64)	Конец временного интервала, в который было зарегистрировано событие (Unix-время)
Фильтр событий (filter)			
aliases	Нет	Array	—
groupBy	Да	Array [String]	Список полей события, по которым нужно сгруппировать события
orderBy	Нет	Array	Сортировка в списке
orderby → field	Нет	String	Поле, по которому нужно сортировать
orderby → sortOrder	Нет	Enum (String)	Порядок сортировки: — ascending — по возрастанию; — descending — по убыванию

Параметр	Обязательный	Тип данных	Описание
select	Да	Array [String]	Список полей события, из которых нужно получить данные
top	Нет	Number (int64)	—
where	Нет	String	Условие фильтрации (предикат фильтра)

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 175. Поля ответа на запрос /api/events/v2/events

Поле	Тип данных	Описание
token	UUID	Токен представления
totalCount	Number (int64)	Количество событий, удовлетворяющих указанным условиям фильтрации
events	Array	Список событий

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/events/v2/events
{
  "filter": {
    "aggregateBy": [],
    "aliases": null,
    "distributeBy": [],
    "groupBy": [],
    "orderBy": [
      {
        "field": "time",
        "sortOrder": "descending"
      }
    ],
```

```
    "select": [
      "time",
      "event_src.host",
      "text",
      "id",
      "uuid",
      "correlation_name",
      "object.type"
    ],
    "top": null,
    "where": ""
  },
  "groupValues": [],
  "timeFrom": 1615826187,
  "timeTo": 1615912587
}
```


Ответ:


```
{
  "events": [
    {
      "time": "2021-03-16T16:01:05.0000000Z",
      "event_src.host": "example.host.ru",
      "text": "Пользователь qa-lisa$ осуществил успешный вход в систему на узле
example.host.ru. Тип входа: Network",
      "id":
"PT_Microsoft_Windows_eventlog_4624_An_account_was_successfully_logged_on",
      "uuid": "00000006-050d-0642-f000-0000255d3663",
      "correlation_name": null,
      "object.type": null,
      "_meta": {
        "id": "00000006-050d-0642-f000-0000255d3663",
        "time": "2021-03-16T16:01:05.0000000Z",
        "assetIds": [
          "1493f164-2240-0001-0000-0000000000c6",
          "1493f163-9100-0001-0000-0000000000a7"
        ],
        "site_alias": "SITE",
        "site_name": "Площадка",
        "site_address": "example.site.ru",
        "site_is_deleted": false
      }
    },
    {
      "time": "2021-03-16T16:01:05.0000000Z",
      "event_src.host": "example.host.ru",
```

```
      "text": "Kerberos-аутентификация пользователя healthmailbox93343d1, запрошенная
с узла 10.0.211.23, прошла успешно. Служба Kerberos на узле example.host.ru выдала
билет проверки подлинности Kerberos (TGT)",
      "id":
"PT_Microsoft_Windows_eventlog_4768_A_Kerberos_authentication_ticket_was_requested",
      "uuid": "00000006-050d-0641-f000-0002255d3663",
      "correlation_name": null,
      "object.type": null,
      "_meta": {
        "id": "00000006-050d-0641-f000-0002255d3663",
        "time": "2021-03-16T16:01:05.0000000Z",
        "assetIds": [
          "1493f1bf-41c0-0001-0000-00000000019f",
          "1493f163-9100-0001-0000-0000000000a7"
        ],
        "site_alias": "SITE",
        "site_name": "Площадка",
        "site_address": "example.site.ru",
        "site_is_deleted": false
      }
    },
  ],
  "totalCount": 125,
  "token": "9176b52b-8e14-4a83-8e24-833af6a9d0b6"
}
```


15.8.2. Привязка событий к инциденту

Запрос для привязки событий к инциденту.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/incidents/events
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 176. Параметры в теле запроса /api/incidents/events

Параметр	Обязательный	Тип данных	Описание
`incidentKey`	Да	String	Ключ инцидента
`events`	Да	Array	Список событий
`events` → `id`	Да	UUID	Системный идентификатор события

Параметр	Обязательный	Тип данных	Описание
`events → timestamp`	Нет	DateTime	Дата и время привязки к инциденту

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации.

Пример

Запрос:

```
POST https://localhost/api/incidents/events
{
  "incidentKey": "INC-1",
  "events": [
    {
      "id": "00000006-0d99-00e2-f000-00009d368550",
      "timestamp": "2021-06-28T09:05:37.000Z"
    },
    {
      "id": "00000006-0d99-00e1-f000-00019d368550",
      "timestamp": "2021-06-28T09:05:35.000Z"
    },
    {
      "id": "00000006-0d99-00e1-f000-00009d368550",
      "timestamp": "2021-06-28T09:05:35.000Z"
    }
  ]
}
```

15.9. Интеграция с активами

С помощью запросов к API вы можете получать информацию о подписчике и его табличных списках, регистрировать подписчика, создавать табличные списки, а также удалять подписчика и его табличные списки.

В этом разделе

15.9.1. Получение списка подписчиков

Запрос для получения списка подписчиков.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/event-tables/v2/subscribers/
```

Параметры строки запроса описаны в таблице ниже.

Таблица 177. Параметры строки запроса /api/event-tables/v2/subscribers/

Параметр	Обязательный	Тип данных	Описание
`types`	Нет	Array [String]	Фильтр по именам типов подписчиков

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 178. Поля ответа на запрос /api/event-tables/v2/subscribers/

Поле	Тип данных	Описание
`subscribers`	Array	Список зарегистрированных подписчиков на изменения табличных списков: — `id` — идентификатор подписчика. Обязательный параметр, тип String; — `name` — имя подписчика. Обязательный параметр, тип String; — `type` — имя типа подписчика. Обязательный параметр, тип String

15.9.2. Регистрация подписчика

Запрос для регистрации подписчика.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/event-tables/v2/subscribers/
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 179. Параметры в теле запроса /api/event-tables/v2/subscribers/

Параметр	Обязательный	Тип данных	Описание
`subscriberInfo`	Да	Object	Информация о подписчике: — `name` — имя подписчика, тип String; — `type` — имя типа подписчика, тип String

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 180. Поля ответа на запрос /api/event-tables/v2/subscribers/

Поле	Тип данных	Описание
`id`	String	Идентификатор подписчика

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.9.3. Получение информации о подписчике

Запрос для получения информации о подписчике.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}
```

URL запроса содержит path-параметр `subscriberId` — уникальный идентификатор подписчика.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 181. Поля ответа на запрос /api/event-tables/v2/subscribers/{subscriberId}

Поле	Тип данных	Описание
`id`	String	Идентификатор подписчика
`name`	String	Имя подписчика
`type`	String	Имя типа подписчика

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.9.4. Удаление подписчика и его табличных списков

Запрос на идемпотентное удаление подписчика и его табличных списков.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}
```

URL запроса содержит path-параметр `subscriberId` — идентификатор подписчика.

Ответ на запрос

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция по созданию группы выполняется.

15.9.5. Получение списка табличных списков подписчика

Запрос для получения списка табличных списков подписчика.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
{tableListId}
```

URL запроса содержит path-параметр `subscriberId` — уникальный идентификатор подписчика.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 182. Поля ответа на запрос /api/event-tables/v2/subscribers/{subscriberId}

Поле	Тип данных	Описание
`lists`	Array	Список табличных списков: — `id` — идентификатор табличного списка, тип String; — `name` — имя табличного списка. Обязательный параметр, тип String; — `isValid` — валиден ли табличный список. Обязательный параметр, тип Bool. Значение по умолчанию: `true`; — `notifications` — перечень уведомлений о состоянии табличного списка: • `errorType` — тип ошибки. Обязательный параметр, тип String; — `ttl` — максимальное время жизни табличного списка, указанное при создании. Пример значения: `90.00:00:00`. Необязательный параметр, тип String ($time-span)

15.9.6. Создание или обновление табличного списка

Запрос для создания или обновления табличного списка.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
```

URL запроса содержит path-параметр `subscriberId` — уникальный идентификатор подписчика.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 183. Параметры в теле запроса /api/event-tables/v2/subscribers/{subscriberId}/tables/

Параметр	Обязательный	Тип данных	Описание
Параметры команды			
`command`	Да	Object	Представление команды на создание или обновление табличного списка: — `name` — имя табличного списка подписчика для идентификации списка в рамках подписчика. Не зависит от регистра. Обязательный параметр, тип String; — `schema` — схема табличного списка. Обязательный параметр, тип Array; • `fields` — набор полей табличного списка (см. ниже). Обязательный параметр, тип Array; • `schemaType` — поле для передачи дискриминатора типа объекта. Обязательный параметр, тип String; — `ttl` — время удаления табличного списка с момента создания или последнего получения данных. Пример значения: `90.00:00:00`. Необязательный параметр, тип String($timespan)
Параметры колонок (fields)			
`sourceFieldName`	Да	String	Имя колонки исходного справочника
`isNullable`	Да	Bool	Может ли содержать null

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 184. Поля ответа на запрос /api/event-tables/v2/subscribers/{subscriberId}/tables/

Поле	Тип данных	Описание
`id`	`String`	Идентификатор табличного списка. Обязательный параметр

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.9.7. Удаление табличных списков подписчика

Запрос для идемпотентного удаления всех табличных списков подписчика.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
```

URL запроса содержит path-параметр `subscriberId` — уникальный идентификатор подписчика.

Ответ на запрос

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция выполняется.

15.9.8. Получение информации о табличном списке подписчика

Запрос для получения информации о табличном списке подписчика.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
```

URL запроса содержит path-параметры `subscriberId` — уникальный идентификатор подписчика и `tableListId` — уникальный идентификатор табличного списка для указанного подписчика.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 185. Поля ответа на запрос /api/event-tables/v2/subscribers/{subscriberId}/tables/

Поле	Тип данных	Описание
`id`	String	Идентификатор табличного списка. Обязательный параметр
`name`	String	Имя подписчика. Обязательный параметр
`isValid`	Bool	Валиден ли табличный список. Обязательный параметр. Значение по умолчанию: `true`
`notifications`	Array	Перечень уведомлений о состоянии табличного списка: — `errorType` — тип ошибки. Обязательный параметр, тип String.
`ttl`	String ($time-span)	Время удаления табличного списка с момента создания или последнего получения данных. Пример значения: `90.00:00:00`. Необязательный параметр

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.9.9. Удаление табличного списка подписчика

Запрос для идемпотентного удаления табличного списка подписчика.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
{tableListId}
```

URL запроса содержит path-параметры `subscriberId` — уникальный идентификатор подписчика и `tableListId` — уникальный идентификатор табличного списка для указанного подписчика.

Ответ на запрос

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция по созданию группы выполняется.

15.9.10. Получение изменений табличного списка

Запрос для получения изменений табличного списка. Если схема табличного списка стала невалидной, изменения табличного списка возможно получить только за период до инвалидации схемы.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/event-tables/v2/subscribers/{subscriberId}/tables/
{tableListId}/changes
```

URL запроса содержит path-параметры `subscriberId` — уникальный идентификатор подписчика и `tableListId` — уникальный идентификатор табличного списка для указанного подписчика.

Параметры строки запроса описаны в таблице ниже.

Таблица 186. Параметры строки запроса /api/event-tables/v2/subscribers/{subscriberId}/changes

Параметр	Обязательный	Тип данных	Описание
`token`	Нет	String	Токен изменений табличного списка. Если параметр отсутствует — выборка из первого элемента
`timeout`	Нет	String ($timespan)	Время максимального ожидания ответа сервера. Пример значения: `90.00:00:00`

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 187. Поля ответа на запрос /api/event-tables/v2/subscribers/{subscriberId}/tables/changes

Поле	Тип данных	Описание
Параметры табличного списка		
`nextToken`	String	Токен для получения набора изменений табличного списка. Необязательный параметр

Поле	Тип данных	Описание
`changes`	Array	Набор изменений табличного списка. Необязательный параметр
`isSchemaValid`	Bool	Валидна ли схема табличного списка. Обязательный параметр
Параметры набора изменений табличного списка		
`added`	Array	Добавленные записи. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3` ... `additionalPropN`. Обязательный параметр
`removed`	Array	Удаленные записи. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3` ... `additionalPropN`. Обязательный параметр

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.10. Работа с учетными записями сканирования

С помощью запросов к API вы можете добавлять учетные записи сканирования с типами «логин — пароль», «пароль» и «сертификат», получать и обновлять информацию об учетных записях с такими типами, а также получать список всех учетных записей и их меток и удалять учетные записи.

В этом разделе

Получение всех учетных записей (см. раздел 15.10.10)

Удаление учетной записи (см. раздел 15.10.11)

Получение списка меток учетных записей (см. раздел 15.10.12)

15.10.1. Добавление учетной записи типа «логин — пароль»

Запрос для добавления учетной записи типа «логин — пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v3/credentials/login_passwords
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 188. Параметры в теле запроса /api/v3/credentials/login_passwords

Параметр	Обязательный	Тип данных	Описание
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
`domain`	Нет	String	Имя домена — если для доступа к источнику используется доменная учетная запись
`login`	Да	String	Логин учетной записи
`password`	Нет	String	Пароль учетной записи

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 189. Поля ответа на запрос /api/v3/credentials/login_passwords

Поле	Тип данных	Описание
`id`	String	Идентификатор учетной записи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.10.2. Получение учетной записи типа «логин — пароль»

Запрос для получения учетной записи типа «логин — пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v3/credentials/login_passwords/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 190. Параметры ответа на запрос /api/v3/credentials/login_passwords/{id}

Параметр	Обязательный	Тип данных	Описание
`type`	Да	String	Тип учетной записи
`id`	Да	String	Идентификатор учетной записи
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора

Параметр	Обязательный	Тип данных	Описание
domain	Нет	String	Имя домена — если для доступа к источнику используется доменная учетная запись
login	Да	String	Логин учетной записи

Возможные коды ошибок и их значения:

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.3. Обновление учетной записи типа «логин — пароль»

Запрос для обновления учетной записи типа «логин — пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/v3/credentials/login_passwords/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры строки запроса описаны в таблице ниже.

Таблица 191. Параметры строки запроса /api/v3/credentials/login_passwords/{id}

Поле	Обязательный	Тип данных	Описание
name	Да	String	Название учетной записи
description	Да	String	Описание
credentialTags	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
domain	Нет	String	Имя домена — если для доступа к источнику используется доменная учетная запись

Поле	Обязательный	Тип данных	Описание
`login`	Да	String	Логин учетной записи
`shouldResetPassword`	Да	Bool	Сбросить ли пароль от учетной записи
`password`	Да	String	Пароль учетной записи

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content). Ответ не содержит полей.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.4. Добавление учетной записи типа «пароль»

Запрос для добавления учетной записи типа «пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v3/credentials/passwords_only
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 192. Параметры в теле запроса /api/v3/credentials/ passwords_only

Параметр	Обязательный	Тип данных	Описание
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
`password`	Да	String	Пароль учетной записи

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 193. Поля ответа на запрос /api/v3/credentials/passwords_only

Поле	Тип данных	Описание
`id`	String	Идентификатор созданной учетной записи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.10.5. Получение учетной записи типа «пароль»

Запрос для получения учетной записи типа «пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v3/credentials/passwords_only/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 194. Параметры ответа на запрос /api/v3/credentials/passwords_only/{id}

Поле	Тип данных	Описание
`type`	String	Тип учетной записи
`id`	String	Идентификатор учетной записи
`name`	String	Название учетной записи
`description`	String	Описание
`credentialTags`	String	Метки учетной записи — если она создается для определенных методов сбора

Возможные коды ошибок и их значения:

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.6. Обновление учетной записи типа «пароль»

Запрос для обновления учетной записи типа «пароль».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/v3/credentials/passwords_only/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры строки запроса описаны в таблице ниже.

Таблица 195. Параметры строки запроса /api/v3/credentials/passwords_only/{id}

Поле	Обязательный	Тип данных	Описание
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
`shouldResetPassword`	Да	Bool	Сбросить ли пароль от учетной записи
`password`	Да	String	Пароль учетной записи

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content). Ответ не содержит полей.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.7. Добавление учетной записи типа «сертификат»

Запрос для добавления учетной записи типа «сертификат».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/v3/credentials/certificates
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 196. Параметры в теле запроса /api/v3/credentials/certificates

Параметр	Обязательный	Тип данных	Описание
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
`login`	Нет	String	Логин учетной записи
`password`	Нет	String	Пароль учетной записи
`certificate`	Да	String	Путь к файлу сертификата

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 197. Поля ответа на запрос /api/v3/credentials/certificates

Поле	Тип данных	Описание
`id`	String	Идентификатор учетной записи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.10.8. Получение учетной записи типа «сертификат»

Запрос для получения учетной записи типа «сертификат».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v3/credentials/certificates/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 198. Параметры ответа на запрос /api/v3/credentials/certificates/{id}

Поле	Тип данных	Описание
`type`	String	Тип учетной записи
`id`	String	Идентификатор учетной записи
`name`	String	Название учетной записи
`description`	String	Описание
`login`	String	Логин учетной записи
`credentialTags`	String	Метки учетной записи — если она создается для определенных методов сбора

Возможные коды ошибок и их значения:

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.9. Обновление учетной записи типа «сертификат»

Запрос для обновления учетной записи типа «сертификат».

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/v3/credentials/certificates/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры строки запроса описаны в таблице ниже.

Таблица 199. Параметры строки запроса /api/v3/credentials/certificates/{id}

Поле	Обязательный	Тип данных	Описание
`name`	Да	String	Название учетной записи
`description`	Да	String	Описание
`credentialTags`	Нет	String	Метки учетной записи — если она создается для определенных методов сбора
`login`	Нет	String	Логин учетной записи
`shouldResetCertificatePassword`	Да	Bool	Сбросить ли пароль от учетной записи
`password`	Да	String	Пароль учетной записи
`certificate`	Да	String	Путь к файлу сертификата

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content). Ответ не содержит полей.

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.10. Получение всех учетных записей

Запрос для получения всех учетных записей в краткой форме.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v3/credentials
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 200. Параметры в теле запроса /api/v3/credentials

Поле	Обязательный	Тип данных	Описание
`credentialTags`	Нет	String	Метки учетных записей — если они созданы для определенных методов сбора

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 201. Параметры ответа на запрос /api/v3/credentials

Поле	Тип данных	Описание
`type`	String	Тип учетной записи
`id`	String	Идентификатор учетной записи
`name`	String	Название учетной записи
`description`	String	Описание
`credentialTags`	String	Метки учетной записи — если она создается для определенных методов сбора

15.10.11. Удаление учетной записи

Запрос для удаления учетной записи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/v3/credentials/{id}
```

URL запроса содержит path-параметр `id` — идентификатор учетной записи.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content). Ответ не содержит полей.

Возможные коды ошибок и их значения:

— 404 (Not Found) — учетная запись с таким идентификатором не найдена.

15.10.12. Получение списка меток учетных записей

Запрос для получения списка меток учетных записей.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v3/credential_tags
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ содержит список меток учетных записей.

15.11. Работа со сканами

С помощью запросов к API вы можете получать метаданные сырого и обработанного скана по идентификатору, получать содержание сырого и обработанного скана, получать коллекцию данных сырых и обработанных сканов, получать пакет информации о сырых и обработанных сканах, а также добавлять новые сканы.

В этом разделе

15.11.1. Добавление нового скана

Запрос для добавления нового скана.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/v1/scans/{scanId}
```

URL запроса содержит path-параметр `scanId` — идентификатор скана.

Параметры строки запроса описаны в таблице ниже.

Таблица 202. Параметры строки запроса /api/v1/scans/{scanId}

Параметр	Обязательный	Тип данных	Описание
`source`	Да	String	Источник сканирования
`scopeId`	Да	String	Идентификатор инфраструктуры
`time`	Да	String	Время сканирования
`jobId`	Нет	String	Идентификатор задачи сканирования
`noTtl`	Нет	Bool	Отключено ли устаревание актива
`replaceEntities`	Нет	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Нет	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 203. Параметры в теле запроса /api/v1/scans/{scanId}

Параметр	Обязательный	Тип данных	Описание
`content`	Да	String	Тело скана

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (Everything is OK. Scan accepted).

Возможные коды ошибок и их значения:

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.2. Получение метаданных скана по идентификатору

Запрос для получения метаданных скана по идентификатору.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/{scanId}
```

URL запроса содержит path-параметр `scanId` — идентификатор скана.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 204. Поля ответа на запрос /api/v1/scans/{scanId}

Поле	Тип данных	Описание
`id`	String	Идентификатор скана
`timeStamp`	String	Время сканирования
`source`	String	Источник сканирования
`scopeId`	String	Идентификатор инфраструктуры
`jobId`	String	Идентификатор задачи сканирования
`noTtl`	Bool	Отключено ли устаревание актива
`replaceEntities`	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`
`type`	String	Тип скана
`isValid`	Bool	Валиден ли скан
`orderedId`	Integer	Порядковый номер скана

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.3. Получение содержания обработанного скана

Запрос для получения содержания валидного обработанного скана в формате XML.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/{scanId}/content
```

URL запроса содержит path-параметр `scanId` — идентификатор скана.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.4. Получение коллекции данных обработанных сканов

Запрос для получения коллекции данных обработанных сканов начиная с указанного времени.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans
```

Параметры строки запроса описаны в таблице ниже.

Таблица 205. Параметры строки запроса /api/v1/scans

Параметр	Обязательный	Тип данных	Описание
`fromDate`	Да	String	Дата, начиная с которой нужно получить данные
`offset`	Да	Integer	Смещение от начала результатов выборки
`limit`	Да	Integer	Размер страницы выдачи
`scopeId`	Нет	String	Идентификатор инфраструктуры сканирования

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 206. Поля ответа на запрос /api/v1/scans

Поле	Тип данных	Описание
`id`	String	Идентификатор скана
`timeStamp`	String	Время сканирования
`source`	String	Источник сканирования
`scopeId`	String	Идентификатор инфраструктуры
`jobId`	String	Идентификатор задачи сканирования
`noTtl`	Bool	Отключено ли устаревание актива
`replaceEntities`	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`
`type`	String	Тип скана
`isValid`	Bool	Валиден ли скан
`orderedId`	Integer	Порядковый номер скана

Возможные коды ошибок и их значения:

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.5. Получение пакета информации об обработанных сканах

Запрос для получения пакета информации об обработанных сканах.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/packages
```

Параметры строки запроса описаны в таблице ниже.

Таблица 207. Параметры строки запроса /api/v1/scans/packages

Параметр	Обязательный	Тип данных	Описание
`fromCheckpoint`	Да	Integer	Токен предыдущего пакета

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 208. Поля ответа на запрос /api/v1/scans/packages

Поле	Тип данных	Описание
`id`	String	Идентификатор скана
`timeStamp`	String	Время сканирования
`source`	String	Источник сканирования
`scopeId`	String	Идентификатор инфраструктуры
`jobId`	String	Идентификатор задачи сканирования
`noTtl`	Bool	Отключено ли устаревание актива
`replaceEntities`	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`
`type`	String	Тип скана
`isValid`	Bool	Валиден ли скан
`orderedId`	Integer	Порядковый номер скана

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.6. Получение метаданных сырого скана по идентификатору

Запрос для получения метаданных сырого скана по идентификатору.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/raw/{scanId}
```

URL запроса содержит path-параметр `scanId` — идентификатор скана.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 209. Поля ответа на запрос /api/v1/scans/raw/{scanId}

Поле	Тип данных	Описание
`id`	String	Идентификатор скана
`timeStamp`	String	Время сканирования
`source`	String	Источник сканирования
`scopeId`	String	Идентификатор инфраструктуры
`jobId`	String	Идентификатор задачи сканирования
`noTtl`	Bool	Отключено ли устаревание актива
`replaceEntities`	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`
`type`	String	Тип скана

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.7. Получение содержания сырого скана в формате XML

Запрос для получения содержания сырого скана в формате XML.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/raw/{scanId}/content
```

URL запроса содержит path-параметр `scanId` — идентификатор скана.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.8. Получение коллекции данных сырых сканов

Запрос для получения коллекции данных сырых сканов начиная с указанного времени.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/raw
```

Параметры строки запроса описаны в таблице ниже.

Таблица 210. Параметры строки запроса /api/v1/scans/raw

Параметр	Обязательный	Тип данных	Описание
fromDate	Да	String	Дата, начиная с которой нужно получить данные
offset	Да	Integer	Смещение от начала результатов выборки
limit	Да	Integer	Размер страницы выдачи
scopeId	Нет	String	Идентификатор инфраструктуры сканирования

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 211. Поля ответа на запрос /api/v1/scans/raw

Поле	Тип данных	Описание
id	String	Идентификатор скана
timeStamp	String	Время сканирования
source	String	Источник сканирования
scopeId	String	Идентификатор инфраструктуры
jobId	String	Идентификатор задачи сканирования
noTtl	Bool	Отключено ли устаревание актива

Поле	Тип данных	Описание
`replaceEntities`	Bool	Замещаются ли все сущности при мердже. Значение по умолчанию — `false`
`createOnly`	Bool	Помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`
`type`	String	Тип скана

Возможные коды ошибок и их значения:

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.11.9. Получение пакета информации о сырых сканах

Запрос для получения пакета информации о сырых сканах.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/v1/scans/raw/packages
```

Параметры строки запроса описаны в таблице ниже.

Таблица 212. Параметры строки запроса /api/v1/scans/raw/packages

Параметр	Обязательный	Тип данных	Описание
`token`	Нет	String	Токен предыдущего пакета

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 213. Поля ответа на запрос /api/v1/scans/raw/packages

Поле	Тип данных	Описание
`token`	String	Токен пакета
`scans`	Array	Перечень описаний сканов: — `id` — идентификатор скана; — `timeStamp` — время сканирования;

Поле	Тип данных	Описание
		— `source` — источник сканирования; — `scopeId` — идентификатор инфраструктуры; — `jobId` — идентификатор задачи сканирования; — `noTtl` — отключено ли устаревание актива; — `replaceEntities` — замещаются ли все сущности при мердже; — `createOnly` — помечен ли скан как предназначенный только для создания актива. Значение по умолчанию — `false`; — `type` — тип скана

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Badly formatted request) — синтаксическая ошибка в запросе.

15.12. Работа с профилями сканирования

С помощью запросов к API вы можете создавать, обновлять и удалять профили сканирования, получать информацию о профиле сканирования, генерировать профили PenTest, проверять наличие уязвимости при сканировании в режиме пентеста, получать список схем профилей сканирования и локализацию параметров схемы модуля, а также удалять из базы ошибки, возникшие при миграции профиля.

В этом разделе

15.12.1. Получение списка профилей сканирования

Запрос для получения списка профилей сканирования с краткой информацией, с возможностью фильтрации по модулю.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_profiles
```

Параметры строки запроса описаны в таблице ниже.

Таблица 214. Параметры строки запроса /api/scanning/v3/scanner_profiles

Параметр	Обязательный	Тип данных	Описание
`moduleId`	Нет	String	Идентификатор модуля

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK) и список профилей сканирования с краткой информацией. Ответ может содержать поля, описанные в таблице ниже.

Таблица 215. Поля ответа на запрос /api/scanning/v3/scanner_profiles

Поле	Тип данных	Описание
`id`	String	Идентификатор профиля сканирования
`name`	String	Имя профиля сканирования
`isSystem`	Bool	Является ли системой
`baseProfileName`	String	Имя базового профиля
`moduleName`	String	Имя модуля
`moduleId`	String	Идентификатор модуля
`output`	String	Путь к профилям сканирования
`agentRequired`	Bool	Необходим ли коллектор
`baseProfileId`	String	Идентификатор базового профиля
`validationErrors`	Array	Ошибки валидации: — `type` — тип ошибки

15.12.2. Создание профиля сканирования

Запрос для создания профиля сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_profiles
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 216. Параметры в теле запроса /api/scanning/v3/scanner_profiles

Параметр	Обязательный	Тип данных	Описание
`profile`	Да	Array	Создаваемый профиль: — `name` — имя профиля, тип string; — `description` — описание профиля; тип string; — `baseProfileId` — идентификатор базового профиля; тип string; — `overrides` — перекрытие параметров профиля. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3`

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created). Ответ может содержать поля, описанные в таблице ниже.

Таблица 217. Поля ответа на запрос api/scanning/v3/scanner_profiles

Поле	Тип данных	Описание
`id`	String	Идентификатор созданного профиля

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 218. Поля ответа на запрос api/scanning/v3/scanner_profiles

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.12.3. Получение расширенной информации о профиле сканирования

Запрос для получения расширенной информации о профиле сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_profiles/{profileId}
```

URL запроса содержит path-параметр `profileId` — идентификатор профиля сканирования.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 219. Поля ответа на запрос /api/scanning/v3/scanner_profiles/{profileId}

Поле	Тип данных	Описание
`output`	String	Путь к профилю сканирования
`description`	String	Описание
`isSystem`	Bool	Является ли системой

Поле	Тип данных	Описание
moduleName	String	Имя модуля
moduleId	String	Идентификатор модуля
baseProfileId	String	Идентификатор базового профиля
baseProfileName	String	Имя базового профиля
validationErrors	Array	Ошибки валидации: — type — тип ошибки, тип string
profile	Array	Профиль. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: additionalProp1, additionalProp2, additionalProp3
overrides	Array	Перекрытие параметров профиля. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: additionalProp1, additionalProp2, additionalProp3

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 220. Поля ответа на запрос api/scanning/v3/scanner_profiles/{profileId}

Поле	Тип данных	Описание
errors	Array	Ошибки: — error — ошибка: • type — тип ошибки; — source — источник ошибки: • displayName — отображаемое имя; • hostName — имя узла; • ipAddresses — IP-адрес; — sensitive — признак значимости ошибки
message	String	Сообщение
code	Integer	Код ошибки

15.12.4. Обновление профиля сканирования

Запрос для обновления профиля сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/scanning/v3/scanner_profiles/{profileId}
```

URL запроса содержит path-параметр `profileId` — идентификатор профиля сканирования.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 221. Параметры в теле запроса /api/scanning/v3/scanner_profiles/{profileId}

Параметр	Обязательный	Тип данных	Описание
`profile`	Да	Array	Создаваемый профиль: — `name` — имя профиля, тип string; — `description` — описание профиля; тип string; — `overrides` — перекрытие параметров профиля. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3`

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 204 (No Content) — скан не найден;

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 222. Поля ответа на запрос api/scanning/v3/scanner_profiles/{profileId}

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.12.5. Удаление профиля сканирования

Запрос для удаления профиля сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/scanning/v3/scanner_profiles/{profileId}
```

URL запроса содержит path-параметр `profileId` — идентификатор профиля сканирования.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 204 (No Content) — профиль сканирования не найден;

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 223. Поля ответа на запрос api/scanning/v3/scanner_profiles/{profileId}

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.12.6. Генерация профиля PenTest

Запрос для генерации профиля PenTest.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_profiles/generate_profile_for_pentest
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 224. Параметры в теле запроса /api/scanning/v3/scanner_profiles/generate_profile_for_pentest

Параметр	Обязательный	Тип данных	Описание
`settings`	Да	Object	Параметры: — `useStandardPorts` — признак использования стандартных портов, тип bool; — `identifiers` — идентификаторы; тип string

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 225. Поля ответа на запрос /api/scanning/v3/scanner_profiles/generate_profile_for_pentest

Поле	Тип данных	Описание
`baseProfileId`	String	Идентификатор базового профиля
`profile`	Array	Профиль сканирования. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3`

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 226. Поля ответа на запрос api/scanning/v3/scanner_profiles/generate_profile_for_pentest

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.12.7. Проверка наличия уязвимости при сканировании в режиме пентеста

Запрос для проверки наличия уязвимости при сканировании в режиме пентеста.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_profiles/pentest/
validate_vulners_for_pentest
```

Параметры строки запроса описаны в таблице ниже.

Таблица 227. Параметры строки запроса /api/scanning/v3/scanner_profiles/ pentest/ validate_vulners_for_pentest

Параметр	Обязательный	Тип данных	Описание
`identifier`	Да	String	Идентификатор

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 228. Поля ответа на запрос /api/scanning/v3/scanner_profiles/ pentest/ validate_vulners_for_pentest

Поле	Тип данных	Описание
`result`	String	Результат поиска уязвимости

15.12.8. Получение списка схем профилей сканирования

Запрос для получения списка схем профилей сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_profile_schemas
```

Параметры строки запроса описаны в таблице ниже.

Таблица 229. Параметры строки запроса /api/scanning/v3/scanner_profile_schemas

Параметр	Обязательный	Тип данных	Описание
`moduleId`	Нет	String	Идентификатор модуля

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 230. Поля ответа на запрос /api/scanning/v3/scanner_profile_schemas

Поле	Тип данных	Описание
`moduleId`	String	Идентификатор модуля
`dataSchema`	Array	Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3`
`formSchema`	Array	Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: `additionalProp1`, `additionalProp2`, `additionalProp3`

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 231. Поля ответа на запрос api/scanning/v3/scanner_profile_schemas

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.12.9. Получение локализации параметров схемы модуля

Запрос для получения локализации параметров схемы модуля.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_profile_schemas/localization
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 232. Поля ответа на запрос /api/scanning/v3/scanner_profile_schemas/localization

Поле	Тип данных	Описание
`additionalProp1`	String	—
`additionalProp2`	String	—
`additionalProp3`	String	—

15.12.10. Удаление из базы ошибок, возникших при миграции профиля

Запрос для удаления из базы ошибок, возникших при миграции профиля.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/scanning/v3/scanner_profiles/{id}/migration_errors
```

URL запроса содержит path-параметр `Id` — идентификатор профиля сканирования.

Ответ на запрос

Если файл с ошибками не создан, сервис возвращает 204 (No Content).

15.13. Управление задачами сканирования

С помощью запросов к API вы можете создавать, обновлять, запускать, останавливать и удалять задачи сканирования, запрашивать отчеты по задачам (в том числе по идентификатору), точки сохранения задач, количество задач (в том числе по инфраструктуре), валидировать задачи, а также сбрасывать закладки для задач и удалять из базы ошибки, возникшие при миграции задач.

В этом разделе

Запрос отчета о всех задачах (см. раздел 15.13.1)

Создание задачи (см. раздел 15.13.2)

15.13.1. Запрос отчета о всех задачах

Запрос для получения отчета о всех задачах.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks
```

Параметры строки запроса описаны в таблице ниже.

Таблица 233. Параметры строки запроса /api/scanning/v3/scanner_tasks

Параметр	Обязательный	Тип данных	Описание
`mainFilter`	Нет	String	Основной фильтр
`additionalFilter`	Нет	String	Дополнительный фильтр

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 234. Поля ответа на запрос /api/scanning/v3/scanner_tasks

Поле	Тип данных	Описание
name	String	Название задачи, задается пользователем
component	Array	Компонент: — type — тип компонента
agent	Array	Коллектор: — id — идентификатор коллектора; — name — имя коллектора
scope	Array	Задача по инфраструктуре: — id — идентификатор задачи; — name — название задачи
profile	Array	Профиль: — id — идентификатор профиля; — name — название профиля
module	Array	Модуль: — id — идентификатор модуля; — name — название модуля
metatransports	Array	Метатранспорты и их параметры. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: additionalProp1, additionalProp2, additionalProp3
status	String	Статус задачи сканирования
created	String	Дата создания задачи в формате ISO 8601
lastRun	String	Дата последнего запуска в формате ISO 8601
lastRunErrorLevel	String	Уровень ошибки последнего запуска

Поле	Тип данных	Описание
include	Array	Признак включения целей сбора данных: — assets — активы: • id — идентификатор актива; • name — имя актива; — targets — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — assetsGroups — группы активов: • id — идентификатор группы активов; • name — название группы активов
exclude	Array	Признак исключения целей сбора данных: — assets — активы: • id — идентификатор актива; • name — имя актива; — targets — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — assetsGroups — группы активов: • id — идентификатор группы активов; • name — название группы активов
isFqdnPriority	Bool	Является ли сканирование активов по FQDN более приоритетным, чем сканирование по IP-адресу
validationState	String	Состояние валидности
lastRunError	Array	Ошибка последнего запуска: — type — тип ошибки; — category — категория ошибки
hostDiscovery	Array	Признак обнаружения узлов до начала сбора данных: — enabled — признак сканирования отвечающих узлов; — profile — профиль: • id — идентификатор профиля; • name — название профиля
hasBookmarks	Bool	Есть ли сохраненное состояние источника

Поле	Тип данных	Описание
credentials	Array	Учетные записи: — id — идентификатор учетной записи; — name — название учетной записи
triggerParameters	Array	Базовые параметры срабатывания по таймеру: — type — тип; — fromDate — дата начала срабатывания; — toDate — дата окончания срабатывания; — isEnabled — разрешено ли срабатывание по таймеру

15.13.2. Создание задачи

Запрос для создания задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_tasks
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 235. Параметры в теле запроса /api/scanning/v3/scanner_tasks

Параметр	Обязательный	Тип данных	Описание
task	Нет	Object	Задача
Параметры задачи			
name	Да	String	Название задачи, задается пользователем
component	Нет	Array	Компонент: — type — тип компонента
agent	Нет	String	Коллектор
scope	Да	Array	Задача по инфраструктуре: — id — идентификатор задачи; — name — название задачи

Параметр	Обязательный	Тип данных	Описание
`profile`	Да	Array	Профиль сканирования: — `id` — идентификатор профиля; — `name` — название профиля
`include`	Да	Array	Признак включения целей сбора данных: — `assets` — активы: • `id` — идентификатор актива; • `name` — имя актива; — `targets` — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — `assetsGroups` — группы активов: • `id` — идентификатор группы активов; • `name` — название группы активов
`exclude`	Да	Array	Признак исключения целей сбора данных: — `assets` — активы: • `id` — идентификатор актива; • `name` — имя актива; — `targets` — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — `assetsGroups` — группы активов: • `id` — идентификатор группы активов; • `name` — название группы активов
`hostDiscovery`	Да	Array	Признак обнаружения узлов до начала сбора данных: — `enabled` — признак сканирования отвечающих узлов; — `profile` — профиль: • `id` — идентификатор профиля; • `name` — название профиля

Параметр	Обязательный	Тип данных	Описание
triggerParameters	Да	Array	Базовые параметры срабатывания по таймеру: — type — тип; — fromDate — дата начала срабатывания; — toDate — дата окончания срабатывания; — isEnabled — разрешено ли срабатывание по таймеру
isFqdnPriority	Нет	Bool	Является ли сканирование сканирование активов по FQDN более приоритетным, чем сканирование по IP-адресу
overrides	Нет	Array	Переопределение параметров профиля. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: additionalProp1, additionalProp2, additionalProp3
savepoints	Нет	Array	Точки сохранения: — target — цель задачи сканирования; — savepoint — точка сохранения

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 236. Поля ответа на запрос /api/scanning/v3/scanner_tasks

Поле	Тип данных	Описание
id	String	Идентификатор задачи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 237. Поля ответа на запрос /api/scanning/v3/scanner_tasks

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; • `category` — категория ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.13.3. Запрос количества задач

Запрос для получения количества задач.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks/count
```

Параметры строки запроса описаны в таблице ниже.

Таблица 238. Параметры строки запроса /api/scanning/v3/scanner_tasks/count

Параметр	Обязательный	Тип данных	Описание
`mainFilter`	Нет	String	Основной фильтр
`additionalFilter`	Нет	String	Дополнительный фильтр

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 239. Поля ответа на запрос /api/scanning/v3/scanner_tasks/count

Поле	Тип данных	Описание
`total`	Integer	Количество задач сканирования

15.13.4. Запрос отчета о всех точках сохранения всех задач

Запрос для получения отчета о всех точках сохранения всех задач.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks/bookmarks
```

Параметры строки запроса описаны в таблице ниже.

Таблица 240. Параметры строки запроса /api/scanning/v3/scanner_tasks/bookmarks

Параметр	Обязательный	Тип данных	Описание
`offset`	Нет	Integer	Смещение страницы от начала выборки
`limit`	Нет	Integer	Количество запрашиваемых элементов выборки

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 241. Поля ответа на запрос /api/scanning/v3/scanner_tasks/bookmarks

Поле	Тип данных	Описание
`items`	Array	Представление конкретной точки сохранения: — `taskId` — идентификатор задачи; — `scopeId` — идентификатор скоупа; — `target` — цель; — `savepoint` — точка сохранения
`totalItems`	Integer	Количество точек сохранения

15.13.5. Запрос количества задач по инфраструктуре

Запрос для получения количества задач по инфраструктуре.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks/scope/{scopeId}/count
```

URL запроса содержит path-параметр scopeId — идентификатор задачи по инфраструктуре.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 242. Поля ответа на запрос /api/scanning/v3/scanner_tasks/scope/{scopeId}/count

Поле	Тип данных	Описание
`total`	Integer	Количество задач по инфраструктуре

15.13.6. Запрос отчета по задаче по идентификатору

Запрос для получения отчета по задаче по идентификатору.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 243. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}

Поле	Тип данных	Описание
`id`	String	Идентификатор задачи
`name`	String	Название задачи

Поле	Тип данных	Описание
include	Array	Признак включения целей сбора данных: — assets — активы: • id — идентификатор актива; • name — имя актива; — targets — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — assetsGroups — группы активов: • id — идентификатор группы активов; • name — название группы активов
exclude	Array	Признак исключения целей сбора данных: — assets — активы: • id — идентификатор актива; • name — имя актива; — targets — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — assetsGroups — группы активов: • id — идентификатор группы активов; • name — название группы активов
isFqdnPriority	Bool	Является ли сканирование сканирование активов по FQDN более приоритетным, чем сканирование по IP-адресу
component	Array	Компонент: — type — тип компонента
agent	Array	Коллектор: — id — идентификатор коллектора; — name — имя коллектора
module	Array	Модуль: — id — идентификатор модуля; — name — название модуля

Поле	Тип данных	Описание
scoup	Array	Задача по инфраструктуре: — id — идентификатор задачи; — name — название задачи
profile	Array	Профиль сканирования: — id — идентификатор профиля; — name — название профиля
completed	String	Дата завершения задачи
status	String	Статус задачи сканирования
created	String	Дата создания задачи
lastRun	String	Дата последнего запуска
hasBookmarks	Bool	Сохранено ли состояние источника
lastRunErrorLevel	String	Уровень ошибки последнего запуска
validationState	String	Состояние валидности
hostDiscovery	Array	Признак обнаружения узлов до начала сбора данных: — enabled — признак сканирования отвечающих узлов; — profile — профиль: • id — идентификатор профиля; • name — название профиля
credentials	Array	Учетные записи: — id — идентификатор учетной записи — name — название учетной записи
triggerParameters	Array	Базовые параметры срабатывания по таймеру: — type — тип; — fromDate — дата начала срабатывания; — toDate — дата окончания срабатывания; — isEnabled — разрешено ли срабатывание по таймеру
overrides	Array	Переопределение параметров профиля. Каждая строка представлена массивом значений столбцов строки в порядке, объявленном в схеме: additionalProp1, additionalProp2, additionalProp3

Возможные коды ошибок и их значения:

404 (Task not found) — задача не найдена.

15.13.7. Обновление задачи

Запрос для обновления задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
PUT <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 244. Параметры в теле запроса /api/scanning/v3/scanner_tasks/{id}

Параметр	Обязательный	Тип данных	Описание
`name`	Да	String	Название задачи, задается пользователем
`component`	Нет	Array	Компонент: — `type` — тип компонента
`agent`	Нет	Array	Идентификатор коллектора
`scope`	Да	Array	Идентификатор задачи по инфраструктуре
`profile`	Да	Array	Идентификатор основного профиля сканирования
`include`	Да	Array	Признак включения целей сбора данных: — `assets` — активы: • `id` — идентификатор актива; • `name` — имя актива; — `targets` — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — `assetsGroups` — группы активов: • `id` — идентификатор группы активов; • `name` — название группы активов

Параметр	Обязательный	Тип данных	Описание
exclude	Да	Array	Признак исключения целей сбора данных: — assets — активы: • id — идентификатор актива; • name — имя актива; — targets — IP-адреса, FQDN или маски подсетей конкретных сетевых адресов; — assetsGroups — группы активов: • id — идентификатор группы активов; • name — название группы активов
hostDiscovery	Да	Array	Признак обнаружения узлов до начала сбора данных: — enabled — признак сканирования отвечающих узлов; — profile — профиль
triggerParameters	Да	Array	Базовые параметры срабатывания по таймеру: — type — тип; — fromDate — дата начала срабатывания; — toDate — дата окончания срабатывания; — isEnabled — разрешено ли срабатывание по таймеру
isFqdnPriority	Нет	Bool	Является ли сканирование активов по FQDN более приоритетным, чем сканирование по IP-адресу
overrides	Нет	Array	Перекрытие настроек профиля
savepoints	Нет	Array	Точки сохранения: — target — цель задачи сканирования; — savepoint — точка сохранения

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 245. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}

Поле	Тип данных	Описание
`id`	String	Идентификатор задачи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 246. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; • `category` — категория ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.13.8. Удаление задачи

Запрос для удаления задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (Успешное удаление).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.13.9. Валидация задачи

Запрос для валидации задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}/validation
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 247. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}/validation

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; • `category` — категория ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.13.10. Сброс закладок для задачи сканирования

Запрос для сброса закладок для задачи сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}/cleanBookmarks
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (Успешный сброс).

15.13.11. Запуск задачи сканирования

Запрос для запуска задачи сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}/start
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 248. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}/start

Поле	Тип данных	Описание
`id`	String	Идентификатор задачи

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе. Ответ может содержать поля, описанные в таблице ниже.

Таблица 249. Поля ответа на запрос /api/scanning/v3/scanner_tasks/{id}/start

Поле	Тип данных	Описание
`errors`	Array	Ошибки: — `error` — ошибка: • `type` — тип ошибки; • `category` — категория ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`message`	String	Сообщение
`code`	Integer	Код ошибки

15.13.12. Остановка запуска задачи

Запрос для остановки запуска задачи сканирования.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}/stop
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (Успешная остановка).

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.13.13. Удаление из базы ошибок, возникших при миграции задачи

Запрос для удаления из базы ошибок, возникших при миграции задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/scanning/v3/scanner_tasks/{id}/migration_errors
```

URL запроса содержит path-параметр `id` — идентификатор задачи.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

15.14. Журналирование действий пользователя

С помощью запросов к API вы можете получать отчеты о действиях пользователей (в том числе по фильтру), дерево категорий действий, возможные значения дополнительного поля, описания всех дополнительных полей для всех приложений, а также регистрировать события пользователя.

В этом разделе

15.14.1. Получение дерева категорий действий

Запрос для получения дерева категорий действий.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/action_categories
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (Success). Ответ может содержать поля, описанные в таблице ниже.

Таблица 250. Поля ответа на запрос /api/ual/v2/action_categories

Поле	Тип данных	Описание
domains	Array	Домены объектов
Параметры доменов объектов		
id	String	Идентификатор домена объекта
name	String	Название домена объекта
applicationId	UUID	Идентификатор приложения
applicationName	String	Название приложения
types	Array	Тип объекта
Параметры типа объекта		
id	String	Идентификатор типа объекта
name	String	Название типа объекта

15.14.2. Получение возможного значения дополнительного поля

Запрос для получения возможного значения дополнительного поля.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/user_actions/applications/
{applicationId}/field_values/{key}
```

URL запроса содержит path-параметры: `applicationId` — идентификатор приложения и `key` — ключ поля.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (Success). Ответ может содержать поля, описанные в таблице ниже.

Таблица 251. Поля ответа на запрос /api/ual/v2/user_actions/applications/{applicationId}/field_values/{key}

Поле	Тип данных	Описание
type	Array	Тип поля

Поле	Тип данных	Описание
Параметры типа поля		
`value`	String	Значение
`relatedKeyValue`	String	Значение поля, к которому относится данное поле

15.14.3. Получение описания всех дополнительных полей для всех приложений

Запрос для получения описания всех дополнительных полей для всех приложений.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/user_actions/additional_fields
```

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (Success). Ответ может содержать поля, описанные в таблице ниже.

Таблица 252. Поля ответа на запрос /api/ual/v2/user_actions/applications/{applicationId}/field_values/{key}

Поле	Тип данных	Описание
`type`	Array	Тип поля
Параметры типа поля		
`applicationId`	String	Идентификатор приложения
`fieldDefinitions`	Array	Список описаний дополнительных полей
Параметры списка описаний дополнительных полей		
`key`	String	Ключ
`localizedName`	String	Локализованное название
`isHidden`	String	Признак сокрытия поля для отображения в списке дополнительных полей. Значение по умолчанию — `false`
`useForFullTextSearch`	Bool	Используются ли значения в полнотекстовом поиске
`filter`	Array	Описание доступного фильтра по данному полю

Поле	Тип данных	Описание
	Параметры фильтра	
`type`	String	Дискриминатор типа фильтра
`selectAllLocalizedName`	String	Локализованное название строки «Выбрать все» при открытии фильтра

15.14.4. Получение отчетов о действиях пользователей

Запрос для получения отчетов о действиях пользователей.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/user_actions
```

Параметры строки запроса описаны в таблице ниже.

Таблица 253. Параметры строки запроса /api/ual/v2/user_actions

Параметр	Обязательный	Тип данных	Описание
`userId`	Да	String	Идентификатор пользователя для фильтрации
`category`	Да	String	Категория в формате domainId.typeId
`orderby`	Да	String	Сортировка, поддерживается только по времени. Пример — `orderby=\"time desc\"`
`query`	Да	String	Строка для поиска, введенная пользователем
`limit`	Да	Integer (int32)	Количество запрашиваемых действий
`offset`	Да	Integer (int32)	Смещение относительно начала списка действий
`timeFrom`	Нет	Long (int64)	Начало периода фильтрации (Unix time). Значение по умолчанию — `0`
`timeTo`	Нет	Long (int64)	Конец периода фильтрации (Unix time). Значение по умолчанию — текущее время

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (Success). Ответ может содержать поля, описанные в таблице ниже.

Таблица 254. Поля ответа на запрос /api/ual/v2/user_actions

Поле	Тип данных	Описание
id	UUID	Идентификатор действия
beginDateTime	Date (date-time)	Время действия в формате ISO 8601
beginTime	Long (int64)	Время действия (Unix time)
beginUtcOffset	String	Сдвиг временной зоны действия относительно UTC
duration	String	Длительность
userId	String	Идентификатор пользователя
userLogin	String	Отображаемое имя пользователя
userDomain	String	Домен пользователя
objectDomainId	String	Идентификатор домена объекта
objectDomain	String	Домен объекта
objectTypeId	String	Идентификатор типа объекта
objectType	String	Тип объекта
objectId	String	Идентификатор объекта
objectDisplayName	String	Отображаемое имя объекта
applicationId	String	Идентификатор приложения
applicationName	String	Название приложения
code	String	Код действия
result	String	Результат действия
failReason	String	Причина ошибки
details	String	Подробное описание изменений
additionalFields	Object	Значение дополнительных полей событий

15.14.5. Получение отчетов о действиях пользователей по фильтру

Запрос для получения отчетов о действиях пользователей по фильтру.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/user_actions
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 255. Параметры в теле запроса /api/ual/v2/user_actions/

Параметр	Обязательный	Тип данных	Описание
`filter`	Да	UserActionFilter	Фильтр сбора событий
Параметры фильтра			
`userIds`	Нет	Array [String]	Массив идентификаторов пользователей
`categories`	Нет	Array [String]	Массив категорий в формате domainId.typeId
`orderby`	Нет	String	Сортировка, поддерживается только по времени. Пример — `orderby=\"time desc\"`
`query`	Нет	String	Строка для поиска, введенная пользователем
`timeFrom`	Нет	Date (date-time)	Дата начала периода фильтрации
`timeTo`	Нет	Date (date-time)	Дата конца периода фильтрации
`additionalFields`	Нет	Array	Фильтры по дополнительным полям
Параметры фильтра по дополнительному полю			
`key`	Да	String	Ключ дополнительного поля
`value`	Да	String	Значение для поиска

Параметры строки запроса описаны в таблице ниже.

Таблица 256. Параметры URL запроса /api/ual/v2/user_actions/

Параметр	Обязательный	Тип данных	Описание
`limit`	Да	Integer	Количество запрашиваемых действий
`offset`	Нет	Integer	Смещение относительно начала списка действий

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (Success). Ответ может содержать поля, описанные в таблице ниже.

Таблица 257. Поля ответа на запрос /api/ual/v2/user_actions/}

Поле	Тип данных	Описание
totalItems	Integer	Общее количество событий, удовлетворяющих условиям запроса
items	Array	Список действий пользователя
Параметры списка действий пользователя		
id	UUID	Идентификатор действия
beginDateTime	Date (date-time)	Время действия в формате ISO 8601
beginTime	Long (int64)	Время действия (Unix time)
beginUtcOffset	String	Сдвиг временной зоны действия от UTC
duration	String	Длительность
userId	String	Идентификатор пользователя
userLogin	String	Отображаемое имя пользователя
userDomain	String	Домен пользователя
objectDomainId	String	Идентификатор домена объекта
objectDomain	String	Домен объекта
objectTypeId	String	Идентификатор типа объекта
objectType	String	Тип объекта
objectId	String	Идентификатор объекта
objectDisplayName	String	Отображаемое имя объекта
applicationId	String	Идентификатор приложения
applicationName	String	Название приложения
code	String	Код действия
result	String	Результат действия
failReason	String	Причина ошибки
details	String	Подробное описание изменений

Поле	Тип данных	Описание
`additionalFields`	Object	Значение дополнительных полей событий

15.14.6. Регистрация событий пользователя

Запрос для регистрации событий пользователя.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>:3334/ptms/api/ual/v2/user_events
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 258. Параметры в теле запроса /api/ual/v2/user_events

Параметр	Обязательный	Тип данных	Описание
`actions`	Да	Array	Список событий пользователя
Параметры списка событий пользователя			
`id`	Да	UUID	Идентификатор действия
`time`	Да	Date (date-time)	Время действия
`type`	Да	String	Тип события
`utcOffset`	Да	String	Сдвиг временной зоны действия от UTC

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created).

15.15. Работа с активами

С помощью запросов к API вы можете сохранить статус завершенной задачи, отменить задачу обновления паспорта активов, запускать задачи на удаление активов, изменение расположения активов в группах, обновление паспортов активов, а также получать состояние этих задач.

В этом разделе

15.15.1. Запуск задачи на удаление активов

Запрос для запуска задачи на удаление активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v1/asset_operations/removeAssets
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 259. Параметры в теле запроса /api/assets_processing/v1/asset_operations/removeAssets

Параметр	Обязательный	Тип данных	Описание
`assetsIds`	Да	String	Идентификаторы удаляемых активов

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created. Возвращаем token для получения состояния задачи). Ответ может содержать поля, описанные в таблице ниже.

Таблица 260. Поля ответа на запрос /api/assets_processing/v1/asset_operations/removeAssets

Поле	Тип данных	Описание
`operationId`	String	Идентификатор операции

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации;

— 503 (ServiceUnavailable) — ошибка доступа.

15.15.2. Получение состояния задачи по удалению активов

Запрос для получения состояния задачи по удалению актовов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v1/asset_operations/removeAssets
```

Параметры строки запроса описаны в таблице ниже.

Таблица 261. Параметры строки запроса /api/assets_processing/v1/asset_operations/removeAssets

Параметр	Обязательный	Тип данных	Описание
`operationId`	Да	String	Идентификатор задачи

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 262. Поля ответа на запрос /api/assets_processing/v1/asset_operations/removeAssets

Поле	Тип данных	Описание
`type`	String	Тип
`totalCount`	Integer	Общее количество активов
`succeedCount`	Integer	Количество успешных удалений активов
`failedCount`	Integer	Количество неуспешных удалений активов

Поле	Тип данных	Описание
errorModel	Integer	Модель ошибок: — errors — ошибки: • error — ошибка: type — тип ошибки; • source — источник; • sensitive — признак значимости ошибки; — message — сообщение; — code — код ошибки
Параметры источника (source)		
displayName	String	Отображаемое имя
hostName	String	Имя узла
ipAddresses	String	IP-адрес

Возможные коды ошибок и их значения:

— 202 (Accepted) — задача еще выполняется;

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.15.3. Запуск задачи на изменение расположения активов в группах

Запрос для запуска задачи на изменение расположения активов в группах.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v1/asset_operations/updateGroupEntries
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 263. Параметры в теле запроса /api/assets_processing/v1/asset_operations/updateGroupEntries

Параметр	Обязательный	Тип данных	Описание
assetsIds	Да	String	Идентификаторы активов
includeInGroups	Да	String	Включенные в группы активы

Параметр	Обязательный	Тип данных	Описание
excludeFromGroups	Да	String	Исключенные из групп активы

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created. Возвращаем token для получения состояния задачи). Ответ может содержать поля, описанные в таблице ниже.

Таблица 264. Поля ответа на запрос /api/assets_processing/v1/asset_operations/updateGroupEntries

Поле	Тип данных	Описание
operationId	String	Идентификатор операции

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации;

— 503 (ServiceUnavailable) — ошибка доступа.

15.15.4. Получение состояния задачи по обновлению расположения активов в группах

Запрос для получения состояния задачи по обновлению расположения активов в группах.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v1/asset_operations/updateGroupEntries
```

Параметры строки запроса описаны в таблице ниже.

Таблица 265. Параметры строки запроса /api/assets_processing/v1/asset_operations/updateGroupEntries

Параметр	Обязательный	Тип данных	Описание
operationId	Да	String	Идентификатор задачи

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 266. Поля ответа на запрос /api/assets_processing/v1/asset_operations/updateGroupEntries

Поле	Тип данных	Описание
`type`	String	Тип
`totalCount`	String	Общее количество
`succeedCount`	String	Количество успешных задач
`failedCount`	String	Количество неуспешных задач
`errorModel`	Array	Данные ошибок
`errorModel` → `errors`	Array	Название ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`errorModel` → `message`	String	Текст ошибки
`errorModel` → `code`	Integer	Код ошибки
`updatedGroups`	Array	Данные обновленной группы
`updatedGroups` → `id`	String	Идентификатор группы
`updatedGroups` → `displayName`	String	Название группы

Возможные коды ошибок и их значения:

— 202 (Accepted) — задача еще выполняется;

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.15.5. Запуск задачи на обновление паспортов активов

Запрос для запуска задачи на обновление паспортов активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
POST <Корневой URL API>/api/assets_processing/v1/asset_operations/updatePassports
```

Тело запроса может содержать параметры, описанные в таблице ниже.

Таблица 267. Параметры в теле запроса /api/assets_processing/v1/asset_operations/updatePassports

Параметр	Обязательный	Тип данных	Описание
`assetsIds`	Да	Array	Идентификаторы активов
`changes`	Да	Array	Изменение: — `ChangeCommand:` • `changeType` — тип изменения
`operationId`	Нет	UUID	Идентификатор операции

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 201 (Created. Возвращаем token для получения состояния задачи). Ответ может содержать поля, описанные в таблице ниже.

Таблица 268. Поля ответа на запрос /api/assets_processing/v1/asset_operations/updatePassports

Поле	Тип данных	Описание
`operationId`	String	Идентификатор операции

Возможные коды ошибок и их значения:

— 400 (Bad request) — синтаксическая ошибка в запросе;

— 401 (Unauthorized) — ошибка аутентификации;

— 503 (ServiceUnavailable) — ошибка доступа.

15.15.6. Получение состояния задачи по обновлению паспорта активов

Запрос для получения состояния задачи по обновлению паспорта активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets_processing/v1/asset_operations/updatePassports
```

Параметры строки запроса описаны в таблице ниже.

Таблица 269. Параметры строки запроса /api/assets_processing/v1/asset_operations/updatePassports

Параметр	Обязательный	Тип данных	Описание
`operationId`	Да	String	Идентификатор задачи

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 270. Поля ответа на запрос /api/assets_processing/v1/asset_operations/updatePassports

Поле	Тип данных	Описание
`type`	String	Тип
`totalCount`	Integer	Общее количество задач
`succeedCount`	Integer	Количество успешных задач
`failedCount`	Integer	Количество неуспешных задач
`errorModel`	Array	Данные ошибок
`errorModel` → `errors`	Array	Название ошибки: — `error` — ошибка: • `type` — тип ошибки; — `source` — источник ошибки: • `displayName` — отображаемое имя; • `hostName` — имя узла; • `ipAddresses` — IP-адрес; — `sensitive` — признак значимости ошибки
`errorModel` → `message`	String	Текст ошибки
`errorModel` → `code`	Integer	Код ошибки

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция еще выполняется;

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.15.7. Отмена задачи обновления паспорта активов

Запрос для отмены задачи по обновлению паспорта активов.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/assets_processing/v1/asset_operations/updatePassports/
{operationId}
```

URL запроса содержит path-параметр `operationId` — идентификатор задачи обновления активов.

Параметры в теле запроса отсутствуют.

Ответ на запрос

Возможные коды ошибок и их значения:

— 202 (Accepted) — операция еще выполняется;

— 400 (Bad request) — синтаксическая ошибка в запросе.

15.15.8. Сохранение статуса завершенной задачи

Запрос для сохранения статуса завершенной задачи.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
DELETE <Корневой URL API>/api/assets_processing/v1/asset_operations/completed/
{operationType}/{operationId}
```

URL запроса содержит path-параметры `operationType` — тип задачи и `operationId` — идентификатор задачи.

Параметры в теле запроса отсутствуют.

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 204 (No Content).

Возможные коды ошибок и их значения:

— 400 (Bad Request) — синтаксическая ошибка в запросе.

15.16. Получение метаданных актива

Запрос для получения метаданных актива.

Для выполнения запроса требуется аутентификация по протоколу OAuth с токеном доступа типа Bearer.

Метод и URL запроса:

```
GET <Корневой URL API>/api/assets/metadata/{type}
```

URL запроса содержит path-параметр `type` — тип модели метаданных.

Параметры строки запроса описаны в таблице ниже.

Таблица 271. Параметры строки запроса /api/assets/metadata/{type}

Параметр	Обязательный	Тип данных	Описание
`type`	Да	String	Тип модели метаданных
`onlyUserProperties`	Нет	Boolean	Используются ли только свойства пользователя

Ответ на запрос

В ответ на успешный запрос сервис возвращает код 200 (OK). Ответ может содержать поля, описанные в таблице ниже.

Таблица 272. Поля ответа на запрос /api/assets/metadata/{type}

Поле	Тип данных	Описание
`name`	String	Имя типа модели метаданных
`root`	String	Корневая группа типа модели метаданных
`kind`	String	Род типа модели метаданных
`title`	String	Заголовок типа модели метаданных
`description`	String	Описание типа модели метаданных
`properties`	String	Свойства типа модели: — `isCollection` — является ли свойство коллекцией; — `name` — имя свойства; — `kind` — род свойства; — `type` — тип свойства; — `title` — наименование свойства; — `description` — описание свойства

Возможные коды ошибок и их значения:

— 304 (Not modified) — запрашиваемая страница не обновлялась с момента последнего обращения;

— 400 (Invalid asset type) — неверный тип актива, синтаксическая ошибка в запросе.

16. Отправка уведомлений через POST-запрос

В MaxPatrol SIEM предусмотрена возможность отправки уведомлений на внешний сервер через POST-запросы. Вы можете настроить отправку таких уведомлений для следующих случаев:

— изменение состава активов в системе или в выбранной группе;

— получение события, удовлетворяющего выбранному фильтру;

— запуск или остановка задачи сбора данных;

— выход параметров потока событий от источника из выбранной группы за пределы допустимых значений;

— появление уведомления о состоянии системы.

Для приема запросов на сервере должно быть установлено специализированное ПО.

Для отправки уведомлений через POST-запросы в веб-интерфейсе MaxPatrol SIEM в разделе **Система** на странице **Уведомления** нужно создать задачу на отправку уведомлений. В параметрах задачи нужно включить отправку уведомления через POST-запрос и указать URL внешнего сервера. Вы можете настроить отправку мгновенных уведомлений или уведомлений за период времени (см. Руководство администратора).

После создания задачи в случае выполнения условия отправки уведомления MaxPatrol SIEM автоматически собирает данные для уведомления и отправляет POST-запрос на внешний сервер. Специализированное ПО на внешнем сервере принимает POST-запрос и по ссылке в запросе получает из MaxPatrol SIEM данные уведомления.

Для приема POST-запросов и получения данных уведомления к ПО выдвигаются следующие требования:

— поддержка обмена данными по протоколам HTTP или HTTPS;

— прием POST-запросов без аутентификации;

— отправка подтверждения с кодом ответа 2xx при приеме POST-запроса;

  Примечание. При отсутствии подтверждения получения запроса от внешнего сервера (с кодом ответа 2xx) POST-запрос будет отправляться повторно до 10 раз с интервалом в минуту.

— получение данных уведомления GET-запросом по ссылке в POST-запросе;

— при получении данных по протоколу HTTPS поддержка проверки подлинности сервера с помощью сертификата SSL.

В этом разделе

16.1. Поля POST-запроса для уведомления

MaxPatrol SIEM отправляет POST-запрос в формате JSON. В зависимости от типа запрос может содержать следующие поля:

- `message_id` — идентификатор POST-запроса.
- `notification_type` — тип POST-запроса: `test` — тестовый запрос, `event` — для мгновенного уведомления, `schedule` — для уведомления за период времени.
- `notification_source` — условие создания уведомления: `AssetStateMetaTrigger` — изменение состава активов в системе; `EventsMetaTrigger` — получение события, удовлетворяющего выбранному фильтру; `EventsMonitoringControlsMetaTrigger` — выход параметров потока событий от источника из выбранной группы за пределы допустимых значений; `GroupContentMetaTrigger` — изменение состава активов в выбранной группе; `HealthMonitoringIssuesMetaTrigger` — появление уведомления о состоянии системы; `ScannerTaskNotificationsMetaTrigger` — запуск или остановка задачи на сбор данных.
- `notification_uid` — идентификатор уведомления.
- `notification_name` — название задачи MaxPatrol SIEM на отправку уведомления.
- `uri` — ссылка на данные уведомления.

  Внимание! Данные уведомления доступны по ссылке в течение 24 часов с момента создания уведомления.
- `schema_uri` — ссылка на схему данных уведомления в формате JSON.
- `event_time_stamp` — дата и время создания мгновенного уведомления (UTC+0).
- `time_interval` — для уведомления за период содержит поля `event_time_start` и `event_time_end` с датами и временем начала и конца периода (UTC+0).
- `time_stamp` — дата и время создания POST-запроса (UTC+0).

Пример тестового POST-запроса:

```
{
  "notification_type":"test",
  "notification_source":"AssetStateMetaTrigger",
  "schema_uri":"https://siem-server.ru:8733/api/assets_triggers/v1/triggers_data/
meta_triggers/AssetStateMetaTrigger/reactions/webhook_notification/schema",
  "time_stamp":"2019-04-15T11:20:49.7031727Z"
}
```

Пример POST-запроса для уведомления за период времени:

```
{
  "message_id":1194,
  "notification_type":"schedule",
  "notification_source":"AssetStateMetaTrigger",
  "notification_uid":"110a1602-d980-0001-0000-000000000006",
```

```
  "notification_name":"1",
"uri":"https://siem-server.ru:8733/api/assets_triggers/v1/triggers_data/
meta_triggers/AssetStateMetaTrigger/reactions/webhook_notification/
110a718e730000010000000000000261",
  "schema_uri":"https://siem-server.ru:8733/api/assets_triggers/v1/triggers_data/
meta_triggers/AssetStateMetaTrigger/reactions/webhook_notification/schema",
  "event_time_stamp":"2019-04-12T09:50:21.4309566Z",
  "time_interval": {
    "event_time_start": "2019-04-12T09:50:21.4309566Z",
    "event_time_end": "2019-04-12T09:55:21.4309566Z"
  },
  "time_stamp":"2019-04-12T09:55:33.0096243Z"
}
```

16.2. Тестирование приема POST-запросов

В качестве примера ПО для приема POST-запросов и получения данных уведомления приводится сценарий на языке Python. В сценарии реализованы следующие возможности:

- При получении POST-запроса в интерфейс командной строки выводится его текст.
- При получении POST-запроса в MaxPatrol SIEM отправляется подтверждение с кодом ответа 2xx.
- По ссылке в POST-запросе из MaxPatrol SIEM скачиваются не более 10 страниц данных уведомления и выводятся в интерфейс командной строки.
- При необходимости по ссылке в POST-запросе из MaxPatrol SIEM скачивается схема данных уведомления и выводится в интерфейс командной строки.
- При получении данных из MaxPatrol SIEM по протоколу HTTPS для проверки подлинности сервера может быть предоставлен сертификат SSL.

Для выполнения сценария на сервере должны быть установлены операционные системы Windows 2012 R2 или Debian 9 и интерпретатор языка Python версии 3.6 или 3.7 с библиотеками Flask и Requests.

> **Примечание.** Для отправки уведомлений на сервер в веб-интерфейсе MaxPatrol SIEM нужно создать задачу на отправку уведомлений через POST-запросы. В параметрах задачи нужно указать URL сервера в виде `http://<IP-адрес или FQDN>/handle`.

► Чтобы настроить прием POST-запросов и получение данных уведомления:

1. Создайте файл с расширением .py и скопируйте в него код сценария.
2. Если требуется, с помощью параметра `HOST` укажите сетевой интерфейс сервера.
3. Если требуется выводить в интерфейс командной строки схему данных уведомления, для параметра `SHOW_SCHEMA` укажите значение `True`.
4. Если требуется, с помощью параметров `HTTP_PORT` и `HTTPS_PORT` измените порты для приема запросов по протоколам HTTP и HTTPS.

5. Если требуется использовать протокол HTTPS, для параметра `USE_HTTPS` укажите значение `True`.

   Внимание! Для проверки подлинности при получении сценарием данных из MaxPatrol SIEM по протоколу HTTPS нужно выпустить сертификат SSL. Файлы сертификата и ключа к нему нужно поместить в одну папку с файлом сценария. С помощью параметров сценария `certfile` и `keyfile` нужно указать имена файлов сертификата и ключа.

6. Сохраните файл.

7. Откройте интерфейс командной строки и запустите сценарий:

   ```
   python <Имя файла сценария>.py
   ```

Прием POST-запросов настроен. Данные уведомлений будут выводиться в интерфейс командной строки.

Код сценария для приема POST-запросов

```
"""pip install flask requests"""
import requests
from pathlib import Path
from flask import Flask, request
HOST = '0.0.0.0'
HTTP_PORT = 10080
HTTPS_PORT = 10443
SHOW_SCHEMA = False
USE_HTTPS = False
cert_dir = Path(__file__).parent
certfile = cert_dir / '<Имя файла сертификата>.crt'
keyfile = cert_dir / '<Имя файла ключа>.pem'
app = Flask('ExampleTriggerService')
@app.route('/handle', methods=['POST'])
def handle():
    data = request.get_json()
    print('INCOMING DATA', data)
    if data['notification_type'] == 'test':
        return print('TEST') or 'test'
    if SHOW_SCHEMA:
        print('GET SCHEMA', data['schema_uri'])
        schema_data = requests.get(data['schema_uri'], verify=False).text
        print('SCHEMA DATA', schema_data)
    page_uri = data['uri']
    for page_num in range(10):
        if not page_uri:
            break
        print('GET PAGE', page_num, page_uri)
        page_data = requests.get(page_uri, verify=False).json()
```

```
        print('PAGE DATA', page_data)
        page_uri = page_data['dataSetInfo']['nextDataSetUri']
    return print('OK') or 'ok'
if __name__ == '__main__':
    if USE_HTTPS:
        import ssl
        context = ssl.SSLContext(ssl.PROTOCOL_TLS)
        # context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2) # для Python версии 3.6
        context.load_cert_chain(str(certfile), str(keyfile))
        port = HTTPS_PORT
    else:
        context = None
        port = HTTP_PORT
    app.run(ssl_context=context, host=HOST, port=port, debug=True)
```

17. Обращение в службу технической поддержки

Техническая поддержка продукта включает в себя следующие услуги:

- решение вопросов эксплуатации продукта, помощь в использовании его функциональных возможностей;
- диагностику сбоев продукта, включая поиск причины сбоя и информирование клиента о найденных проблемах;
- разрешение проблем с продуктом, предоставление решений или возможностей обойти проблему с сохранением всей необходимой производительности;
- устранение ошибок в продукте (в рамках выпуска обновлений к продукту).

Вы можете получать техническую поддержку на портале.

Этот раздел содержит информацию о способах и условиях получения технической поддержки.

В этом разделе

17.1. Техническая поддержка на портале

Портал предоставляет вам возможность создавать запросы на техническую поддержку.

Вы можете создать учетную запись на портале, используя адреса электронной почты, расположенные на официальном домене вашей организации. Вы также можете указывать другие адреса электронной почты для учетной записи в качестве дополнительных. Для оперативной связи укажите в профиле учетной записи название вашей организации и контактный телефон.

Портал содержит статьи базы знаний, новости обновлений продуктов Positive Technologies, ответы на часто задаваемые вопросы пользователей. Для доступа к базе знаний и всем новостям нужно создать на портале учетную запись.

Техническая поддержка на портале предоставляется на русском и английском языках.

17.2. Время работы службы технической поддержки

На портале технической поддержки вы можете круглосуточно создавать и обновлять запросы, читать новости продуктов и пользоваться базой знаний.

17.3. Как служба технической поддержки работает с запросами

При получении вашего запроса специалист службы технической поддержки классифицирует его (присваивает запросу тип и уровень значимости) и выполняет дальнейшие шаги по выполнению запроса.

В этом разделе

17.3.1. Предоставление информации для технической поддержки

При обращении за технической поддержкой по первому требованию специалиста Positive Technologies нужно предоставить:

- номер лицензии на использование продукта;
- файлы журналов и другие наборы диагностических данных, хранящихся в продукте;
- снимки экрана;
- результаты выполнения рекомендаций специалиста технической поддержки;
- каналы для удаленного доступа к продукту (по взаимному согласованию оптимального канала диагностики).

Positive Technologies не несет обязательств по оказанию технической поддержки в случае отказа предоставить указанную выше информацию.

Если информация по обращению не предоставлена в течение значительного времени (от двух недель с момента последней активности), специалист технической поддержки имеет право считать ваше обращение неактуальным и, уведомив вас, закрыть запрос.

17.3.2. Типы запросов

Специалист технической поддержки относит ваш запрос к одному из следующих типов.

Вопросы по установке, повторной установке и предстартовой настройке продукта

Подразумевается помощь в подготовке продукта к работе, ответы на вопросы на данном этапе эксплуатации продукта. Техническая поддержка по этим вопросам доступна в течение 30 дней с момента активации продукта.

Вопросы по администрированию и настройке продукта

Включают в себя вопросы, возникающие в процессе эксплуатации продукта, рекомендации по оптимизации и настройке параметров продукта.

Восстановление работоспособности продукта

В случае критического сбоя и потери доступа к основной функциональности продукта специалист Positive Technologies оказывает помощь в восстановлении работоспособности продукта. Восстановление заключается либо в помощи по установке продукта заново с потенциальной потерей накопленных до сбоя данных, либо в откате продукта на доступную резервную копию (резервное копирование должно быть настроено заблаговременно). Positive Technologies не несет ответственность за потерю данных в случае неверно настроенного резервного копирования.

Обновление продукта

Positive Technologies поставляет пакеты обновления в течение срока обновления, указанного в лицензии на продукт.

Positive Technologies не несет ответственности за проблемы, возникшие при нарушении регламентированного процесса обновления.

Устранение дефектов продукта

Если по результатам диагностики обнаружен дефект продукта, Positive Technologies обязуется предпринять разумные усилия по предоставлению обходного решения (если возможно), а также включить исправление дефекта в ближайшие возможные обновления продукта.

17.3.3. Время реакции и приоритизация запросов

Время реакции на запрос рассчитывается с момента получения запроса до первичного ответа специалиста технической поддержки с уведомлением о взятии запроса в работу.

Время обработки запроса рассчитывается с момента отправки уведомления о взятии вашего запроса в работу до предоставления описания дальнейших шагов по устранению проблемы либо классификации вопроса, указанного в запросе, как дефекта ПО и передачи запроса ответственным лицам для исправления дефекта.

Время реакции и время обработки зависят от указанного вами уровня значимости запроса (см. таблицу 273).

Специалист службы технической поддержки оставляет за собой право переопределять уровень значимости запроса по приведенным ниже критериям. Указанные сроки являются целевыми и подразумевают стремление и разумные усилия исполнителя для их соблюдения, но возможны отклонения от данных сроков по объективным причинам.

Таблица 273. Время реакции на запрос и время его обработки

Уровень значимости запроса	**Критерии значимости запроса**	**Время реакции на запрос**	**Время обработки запроса**
Критический	Аварийные сбои, полностью препятствующие штатной работе продукта (исключая первоначальную установку) либо оказывающие критическое влияние на бизнес	До 4 часов	Не ограничено
Высокий	Сбои, затрагивающие часть функциональности продукта и проявляющиеся в любых условиях эксплуатации либо оказывающие значительное влияние на бизнес	До 24 часов	Не ограничено
Обычный	Сбои, проявляющиеся в специфических условиях эксплуатации продукта либо не оказывающие значительного влияния на бизнес	До 24 часов	Не ограничено
Низкий	Вопросы информационного характера либо сбои, не влияющие на эксплуатацию продукта	До 24 часов	Не ограничено

Указанные часы относятся только к рабочему времени специалистов технической поддержки (времени обработки запроса).

17.3.4. Выполнение работ по запросу

По мере выполнения работ по вашему запросу специалист технической поддержки сообщает вам:

— о диагностике проблемы и ее результатах;

— о поиске решения или возможности обойти причины возникновения проблемы;

— о планировании и выпуске обновления продукта (если требуется для устранения проблемы).

Если по итогам обработки запроса необходимо внести изменения в продукт, Positive Technologies включает работы по исправлению в ближайшее возможное плановое обновление продукта (в зависимости от сложности изменений).

Работы по запросу считаются выполненными, если:

— предоставлено решение или возможность обойти проблему, не влияющая на производительность и критически важную функцию продукта;

— диагностирован дефект продукта, собрана техническая информация о дефекте и условиях его воспроизведения; исправление дефекта запланировано к выходу в рамках планового обновления продукта;

— проблема вызвана программными продуктами или оборудованием сторонних производителей, не подпадающих под гарантийные обязательства по продукту;

— проблема классифицирована как неподдерживаемая.

Приложение A. Типы данных

При использовании функций и операторов языка XP выполняется приведение типов данных. Кроме того, нужно учитывать тип данных при заполнении полей событий (см. раздел 10.3).

Таблица 274. Используемые типы данных

Тип	Описание	Пример
Bool	Логическое значение	`True`, `False`
Buffer	Массив байтов	`[0x68, 0x65, 0x6c, 0x6c, 0x6f, 0x20, 0x77, 0x6f, 0x72, 0x6c, 0x64]`
DateTime	Время в формате ISO 8601: YYYY-MM-DD'T'HH:MM:SS'Z'	`2016-04-20T23:04:38Z`
Enum	Один из элементов предопределенного списка значений	—
IPAddress	IP-адрес стандарта IPv4 или IPv6	`192.0.2.235`, `1080:0:0:0:8:800:200C:417A`
KeyValue	Ассоциативный массив пар «ключ — значение»	`{"Красный":"Каждый", "Оранжевый":"Охотник", "Желтый":"Желает"}`
List	Список. Один список может содержать элементы разных типов	`["Порт ", 22, " открыт"]`
MACAddress	MAC-адрес	`00:53:00:B8:DF:B8`
Network	Адрес подсети с маской в формате CIDR	`192.0.2.0/24`
Null	Отсутствие данных	`null`
Number	Целое число от –223372036854775808 до 9223372036854775807	`-3`, `0`, `12`
String	Строка	`"Порт 22 открыт"`
StringList	Список, элементы которого имеют тип данных String	`["Красный", "Оранжевый", "Желтый", "Зеленый"]`
UUID	Идентификатор в формате RFC 4122 — 16-байтный (128-битный) номер	`123e4567-e89b-12d3-a456-426655440000`
UUIDList	Список идентификаторов UUID	`["00000005-9d7c-011d-f000-0001e5c6e294", "00000005-9d7c-011d-f000-0001e5c6e295"]`

Приложение Б. Схема полей событий

Схема полей событий представляет собой набор полей и их допустимых значений для сохранения информации о событиях на всех этапах их обработки. Столбец «Обязательное» указывает на необходимость обязательно заполнить поле в правиле нормализации или корреляции.

Таблица 275. Схема полей событий

Поле	Обязательное	Тип данных	Описание
Поля корреляционного события			
`alert.context`	Нет	String	Дополнительная информация о событии ИБ
`alert.key`	Нет	String	Дополнительная техническая информация о событии ИБ
`assets`	Автоматически	StringList	IP-адреса и имена узлов, вовлеченных в инцидент. Эти узлы указаны в событиях, на основании которых зарегистрировано корреляционное событие
`attacking_assets`	Автоматически	StringList	IP-адреса и имена узлов, являющихся источниками атаки. Эти узлы указаны в событиях, на основании которых зарегистрировано корреляционное событие
`correlation_name`	Нет	String	Название правила корреляции (уникальное в рамках всего набора правил)
`correlation_type`	При корреляции	Enum	Результат выполнения правила корреляции: `event` — корреляционное событие; `incident` — корреляционное событие и инцидент
`count.subevents`	Нет	Number	Количество событий, на основании которых зарегистрировано агрегированное или корреляционное событие

Поле	Обязательное	Тип данных	Описание
detect	Нет	String	Описание атаки
labels	Нет	String	Метки события ИБ
subevents	Автоматически	UUIDList	Список идентификаторов событий, на основании которых зарегистрировано агрегированное или корреляционное событие
subevents.time	Автоматически	StringList	Время регистрации событий, на основании которых зарегистрировано агрегированное или корреляционное событие
Поля категоризации событий			
category.generic	Нет	String	Уровень категоризации generic (см. приложение Д)
category.high	Нет	String	Уровень категоризации high
category.low	Нет	String	Уровень категоризации low
Адресные поля источника события			
assigned_src_host	Нет	String	Полное доменное имя, соответствующее IP-адресу, назначенному источнику
assigned_src_ip	Нет	IPAddress	IP-адрес, назначенный источнику. Адрес может быть назначен, например, DHCP-сервером, при трансляции адресов NAT или при подключении к VPN
assigned_src_port	Нет	Number	Порт, назначенный источнику при трансляции NAT
src.asset	Нет	UUID	Идентификатор актива, являющегося субъектом взаимодействия
src.fqdn	Нет	String	Полное доменное имя узла-источника при взаимодействии сторон (в формате FQDN)

Поле	Обязательное	Тип данных	Описание
src.geo.asn	Нет	Number	Номер автономной системы узла-источника при взаимодействии сторон
src.geo.city	Нет	String	Город узла-источника при взаимодействии сторон
src.geo.country	Нет	String	Страна узла-источника при взаимодействии сторон
src.geo.org	Нет	String	Организация, которой принадлежит узел-источник, при взаимодействии сторон
src.host	Нет	String	Заполняется автоматически на основании одного из заполненных полей — src.ip или src.hostname
src.hostname	Нет	String	Имя узла источника при взаимодействии сторон (без имени домена)
src.ip	Нет	IPAddress	IP-адрес источника при взаимодействии сторон (IPv4 или IPv6)
src.mac	Нет	MACAddress	MAC-адрес источника при взаимодействии сторон
src.port	Нет	Number	Порт источника при взаимодействии сторон
Поля, описывающие источник события			
event_src.asset	Нет	UUID	Идентификатор актива, являющегося источником события
event_src.category	При нормализации	Enum	Категория источника события (см. приложение Г), например IDS/IPS, Firewall, Mail server
event_src.fqdn	Нет	String	Полное доменное имя узла-источника события (в формате FQDN)
event_src.host	Автоматически	String	Узел источника. Заполняется автоматически на основании полей event_src.hostname, event_src.ip, recv_ipv4, recv_ipv6

Поле	Обязательное	Тип данных	Описание
event_src.hostname	Нет	String	Имя узла источника события (без имени домена)
event_src.id	Нет	String	Уникальный идентификатор источника события
event_src.ip	Нет	IPAddress	IP-адрес источника события (IPv4 или IPv6)
event_src.rule	Нет	String	Название правила, службы или приложения, благодаря которым на источнике зарегистрировано событие
event_src.subsys	Нет	String	Подсистема источника события, в которой генерируются события этого типа
event_src.title	При нормализации	String	Название приложения (источника), которое зарегистрировало событие. Указывается в нижнем регистре без пробелов, например windows, asa, ios
event_src.vendor	При нормализации	String	Производитель приложения (источника), которое зарегистрировало событие. Указывается в нижнем регистре без пробелов, например microsoft, cisco
Адресные поля получателя			
assigned_dst_host	Нет	String	Полное доменное имя, соответствующее IP-адресу, который назначен узлу получателя
assigned_dst_ip	Нет	IPAddress	IP-адрес, назначенный узлу получателя при трансляции NAT
assigned_dst_port	Нет	Number	Порт, назначенный узлу получателя при трансляции NAT
dst.asset	Нет	UUID	Идентификатор актива, являющегося объектом взаимодействия

Поле	Обязательное	Тип данных	Описание
dst.fqdn	Нет	String	Полное доменное имя узла назначения при взаимодействии сторон (в формате FQDN)
dst.geo.asn	Нет	Number	Номер автономной системы узла назначения при взаимодействии сторон
dst.geo.city	Нет	String	Город узла назначения при взаимодействии сторон
dst.geo.country	Нет	String	Страна узла назначения при взаимодействии сторон
dst.geo.org	Нет	String	Организация, которой принадлежит узел назначения, при взаимодействии сторон
dst.host	Нет	String	Заполняется автоматически на основании одного из заполненных полей — dst.ip или dst.hostname
dst.hostname	Нет	String	Имя узла назначения при взаимодействии сторон (без имени домена)
dst.ip	Нет	IPAddress	IP-адрес (IPv4 или IPv6) назначения при взаимодействии сторон
dst.mac	Нет	MACAddress	MAC-адрес назначения при взаимодействии сторон
dst.port	Нет	Number	Порт назначения
Параметры взаимодействия			
action	Обязательно	Enum	Характер воздействия на объект (см. приложение В)
chain_id	Нет	String	Идентификатор последовательности событий
direction	Нет	String	Направление взаимодействия сторон (например, направление сетевого соединения, направление доверительных отношений между доменами)

Поле	Обязательное	Тип данных	Описание
duration	Нет	Number	Продолжительность процесса в секундах
importance	Обязательно	Enum	Уровень важности события для информационной безопасности. Возможные значения: — info — информационное событие, — low — низкая, — medium — средняя, — high — высокая
logon_auth_method	Нет	Enum	Способ подключения при аутентификации на источнике: local — локальный; remote — удаленный
logon_service	Нет	String	Имя службы (модуля), через которую выполнена аутентификация на источнике, или название ПО (терминала), используемого при взаимодействии с источником
logon_type	Нет	Number	Тип аутентификации на источнике
protocol	Нет	String	Протокол передачи данных
protocol.layer7	Нет	String	Протокол уровня приложения (по модели OSI)
reason	Нет	String	Описание причины, по которой произошло событие (например, обрыв соединения, неудачная попытка начать сессию, неудачная аутентификация)
start_time	Нет	DateTime	Время регистрации первого события в последовательности
status	Обязательно	Enum	Конечный результат воздействия

Поле	Обязательное	Тип данных	Описание
Поля, описывающие субъект при взаимодействии			
subject	Нет	Enum	Субъект воздействия
subject.account.contact	Нет	String	Контакты пользователя учетной записи, например телефон или адрес электронной почты
subject.account.dn	Нет	String	Отличительное имя учетной записи (Distinguished Name)
subject.account.domain	Нет	String	Домен учетной записи
subject.account.fullname	Нет	String	Данные, указанные пользователем учетной записи, например его имя и фамилия
subject.account.group	Нет	String	Имена групп, в которые входит учетная запись, или их идентификаторы, перечисленные через « | »
subject.account.id	Нет	String	Идентификатор учетной записи
subject.account.name	Нет	String	Логин учетной записи
subject.account.privileges	Нет	String	Права и привилегии учетной записи
subject.account.session_id	Нет	String	Идентификатор сессии, например Logon ID
subject.domain	Нет	String	Название домена субъекта
subject.group	Нет	String	Имя группы, в которую входит субъект
subject.id	Нет	String	Идентификатор субъекта
subject.name	Нет	String	Имя субъекта
subject.privileges	Нет	String	Привилегии субъекта

Поле	Обязательное	Тип данных	Описание
subject.process.cmdline	Нет	String	Команда запуска процесса (включая аргументы)
subject.process.cwd	Нет	String	Рабочая папка процесса
subject.process.fullpath	Нет	String	Полный путь к исполняемому файлу процесса (объединение строк из полей subject.process.path и subject.process.name)
subject.process.guid	Нет	String	GUID процесса
subject.process.hash	Нет	String	Названия алгоритмов и хеш-суммы исполняемого файла процесса в виде: <Функция 1>:<Сумма 1> ... <Функция N>:<Сумма N>. Блоки должны быть разделены пробелами
subject.process.id	Нет	String	PID процесса
subject.process.meta	Нет	String	Метаданные исполняемого файла процесса в виде: Description:<Описание> | Product:<Название> | Company:<Производитель>
subject.process.name	Нет	String	Имя исполняемого файла процесса
subject.process.original_name	Нет	String	Имя исполняемого файла процесса, указанное разработчиком в метаданных
subject.process.parent.cmdline	Нет	String	Команда запуска родительского процесса (включая аргументы)
subject.process.parent.fullpath	Нет	String	Полный путь к исполняемому файлу родительского процесса (объединение строк из полей subject.process.parent.path и subject.process.parent.name)
subject.process.parent.guid	Нет	String	GUID родительского процесса

Поле	Обязательное	Тип данных	Описание
subject.process.parent.hash	Нет	String	Названия алгоритмов и хеш-суммы исполняемого файла родительского процесса в виде: <Функция 1>:<Сумма 1> ... <Функция N>:<Сумма N>. Блоки должны быть разделены пробелами
subject.process.parent.id	Нет	String	PID родительского процесса
subject.process.parent.name	Нет	String	Имя исполняемого файла родительского процесса
subject.process.parent.path	Нет	String	Путь к исполняемому файлу родительского процесса (без имени)
subject.process.path	Нет	String	Путь к исполняемому файлу процесса (без имени)
subject.process.version	Нет	String	Версия исполняемого файла процесса
subject.state	Нет	String	Состояние субъекта
subject.type	Нет	String	Тип субъекта
subject.version	Нет	String	Версия субъекта
Поля, описывающие объект при взаимодействии			
object	Обязательно	Enum	Объект воздействия
object.account.contact	Нет	String	Контактная информация пользователя учетной записи (номер телефона или адрес электронной почты)
object.account.dn	Нет	String	Уникальное в рамках Active Directory имя учетной записи (Distinguished Name)
object.account.domain	Нет	String	Домен учетной записи

Поле	Обязательное	Тип данных	Описание
object.account.fullname	Нет	String	Данные, указанные пользователем учетной записи, например его имя и фамилия
object.account.group	Нет	String	Перечень групп, в которых состоит учетная запись. Идентификаторы или названия групп должны быть разделены вертикальной чертой с пробелами « | »
object.account.id	Нет	String	Идентификатор учетной записи
object.account.name	Нет	String	Логин учетной записи
object.account.privileges	Нет	String	Права и привилегии учетной записи
object.account.session_id	Нет	String	Идентификатор сессии учетной записи, например Logon ID
object.domain	Нет	String	Имя домена объекта
object.fullpath	Нет	String	Полный путь к объекту (объединение строк из полей object.path и object.name)
object.group	Нет	String	Имя группы, в которую входит объект
object.hash	Нет	String	Хеш-сумма объекта
object.id	Нет	String	Идентификатор объекта
object.name	Нет	String	Имя объекта
object.new_value	Нет	String	Конечное значение измененного свойства объекта
object.num_value	Нет	Number	Исходное значение измененного свойства объекта (численное значение)
object.path	Нет	String	Путь к объекту

Поле	Обязательное	Тип данных	Описание
object.process.cmdline	Нет	String	Командная строка запуска процесса
object.process.cwd	Нет	String	Рабочая папка процесса
object.process.fullpath	Нет	String	Полный путь к исполняемому файлу процесса (объединение строк из полей object.process.path и object.process.name)
object.process.guid	Нет	String	GUID процесса
object.process.hash	Нет	String	Названия алгоритмов и хеш-суммы исполняемого файла процесса в виде: <Функция 1>:<Сумма 1> ... <Функция N>:<Сумма N>. Блоки должны быть разделены пробелами
object.process.id	Нет	String	Идентификатор процесса (PID)
object.process.meta	Нет	String	Метаданные исполняемого файла процесса в виде: Description:<Описание> | Product:<Название> | Company:<Производитель>. Блоки должны быть разделены вертикальной чертой с пробелами « | »
object.process.name	Нет	String	Имя исполняемого файла процесса
object.process.original_name	Нет	String	Имя исполняемого файла процесса, указанное разработчиком и полученное из метаданных
object.process.parent.cmdline	Нет	String	Команда строка запуска родительского процесса
object.process.parent.fullpath	Нет	String	Полный путь к исполняемому файлу родительского процесса (объединение строк из полей object.process.parent.path и object.process.parent.name)
object.process.parent.guid	Нет	String	GUID родительского процесса

Поле	Обязательное	Тип данных	Описание
object.process.parent.hash	Нет	String	Названия алгоритмов и хеш-суммы исполняемого файла родительского процесса в виде: <Функция 1>:<Сумма 1> ... <Функция N>:<Сумма N>. Блоки должны быть разделены пробелами
object.process.parent.id	Нет	String	Идентификатор родительского процесса (PID)
object.process.parent.name	Нет	String	Имя исполняемого файла родительского процесса
object.process.parent.path	Нет	String	Путь к папке, в которой находится исполняемый файл родительского процесса
object.process.path	Нет	String	Путь к папке, в которой находится исполняемый файл процесса
object.process.version	Нет	String	Версия исполняемого файла процесса
object.property	Нет	String	Изменяемое свойства объекта
object.query	Нет	String	Выполненный запрос
object.state	Нет	String	Состояние объекта
object.storage.fullpath	Нет	String	Полный путь к объекту в хранилище (объединение строк из полей object.storage.name и object.storage.path)
object.storage.id	Нет	String	Идентификатор или имя хранилища, в котором расположен объект (например, название общей папки или общего диска)
object.storage.name	Нет	String	Имя объекта в хранилище (если отличается от публичного)
object.storage.path	Нет	String	Путь к объекту в хранилище (если отличается от публичного)
object.type	Нет	String	Тип объекта

Поле	Обязательное	Тип данных	Описание
object.value	Нет	String	Значение изменяемого свойства объекта
object.vendor	Нет	String	Производитель объекта
object.version	Нет	String	Версия объекта
Точка сбора			
recv_asset	Нет	UUID	Идентификатор актива, являющегося передатчиком события
recv_host	Автоматически	String	Имя узла, от которого получено событие
recv_ipv4	Автоматически	IPAddress	IPv4-адрес узла, от которого получено событие
recv_ipv6	Автоматически	IPAddress	IPv6-адрес узла, от которого получено событие
recv_time	Автоматически	DateTime	Время сбора события MP 10 Collector (в часовом поясе UTC+0)
Информационные поля			
asset_ids	Автоматически	UUIDList	Перечень идентификаторов в хранилище активов
count	Автоматически	Number	Количество агрегированных событий
count.bytes	Нет	Number	Суммарное количество байтов в ответе сервера
count.bytes_in	Нет	Number	Количество входящих байтов
count.bytes_out	Нет	Number	Количество исходящих байтов
count.packets	Нет	Number	Суммарное количество пакетов
count.packets_in	Нет	Number	Количество входящих пакетов
count.packets_out	Нет	Number	Количество исходящих пакетов

Поле	Обязательное	Тип данных	Описание
datafield1 ... datafield10	Нет	String	Данные, не укладывающиеся в схему полей событий
external_link	Нет	String	Ссылка на внешний ресурс с подробными сведениями о событии
incorrect_time	Автоматически	Bool	Корректность времени события. Если в поле original_time указано время регистрации события более суток назад и поле historical = false, то возможно, что время события указано неверно (incorrect_time = true). В иных случаях время считается корректным (incorrect_time = false)
interface	Нет	String	Название сетевого интерфейса, через который происходит взаимодействие
msgid	Нет	String	Идентификатор типа события, уникальный в рамках событий одного источника
nas_fqdn	Нет	String	Полное доменное имя (FQDN) NAS-сервера
numfield1 ... numfield5	Нет	Number	Числовые данные, не укладывающиеся в схему полей событий
nas_ip	Нет	IPAddress	IP-адрес NAS-сервера
original_time	Автоматически	DateTime	Время регистрации события на источнике

Поле	Обязательное	Тип данных	Описание
tcp_flag	Нет	String	Флаг в заголовке сегмента TCP (ACK, URG, FIN). Значение поля может состоять из нескольких флагов
time	Обязательно при нормализации. Автоматически при корреляции	DateTime	Время события в MaxPatrol SIEM (в часовом поясе UTC+0). Обычно time = original_time, для ненормализованных событий time = recv_time. Кроме того, если в поле original_time указано время регистрации события более суток назад и поле historical = false, то incorrect_time = true и time = recv_time. **Примечание.** Суточный интервал устаревания события может быть изменен в конфигурационном файле siem.conf в объекте normalizer в параметре expected_event_time_deviation
Сервисные поля			
agent_id	Автоматически	UUID	Идентификатор MP 10 Collector, получившего событие
aggregation_name	Автоматически	String	Название правила агрегации
body	Автоматически	String	Текст необработанного события в кодировке UTF-8
event_type	Автоматически	String	Не используется
generator	Автоматически	String	Не используется
generator.type	Автоматически	Enum	Название компонента, зарегистрировавшего событие: — logcollector — служба нормализации или агрегации, — correlationengine — служба корреляции,

Поле	Обязательное	Тип данных	Описание
			— `srcmonitoring` — мониторинг источников, — `iocengine` — модуль batcheventprocessing
`generator.version`	Автоматически	String	Версия службы, зарегистрировавшей событие
`historical`	Автоматически	Bool	Совпадает со значением параметра `historical`, указанным в профиле при сборе события: `true` — устаревшее событие, `false` — обычное событие
`id`	Обязательно	String	Идентификатор события в виде: — для правила нормализации — `<Пространство имен>_<Производитель>_<Название источника>_<Транспорт>_<Тип события>`; — для правила корреляции — `<Пространство имен>_SIEM_<Название правила корреляции>`
`input_id`	Автоматически	UUID	Идентификатор источника события
`job_id`	Автоматически	UUID	Идентификатор подзадачи MP 10 Collector на сбор событий
`mime`	Автоматически	String	MIME-тип данных необработанного события: — `text/plain` — простой текст, — `application/json` — формат JSON, — `application/x-pt-eventlog` — событие журнала Windows,

Поле	Обязательное	Тип данных	Описание
			— `text/csv` — ассоциативный массив в формате JSON, — `text/xml` — XML
`normalized`	Автоматически	Bool	Статус нормализации события: `true` — нормализованное или корреляционное событие, `false` — необработанное событие
`origin_app_id`	Автоматически	UUID	Идентификатор приложения, которое получило или зарегистрировало событие
`primary_siem_app_id`	Автоматически	UUID	Идентификатор приложения, которое обработало событие
`remote`	Автоматически	Bool	Статус репликации события: `true` — событие было зарегистрировано на одной площадке и передано на другую по правилу репликации, `false` — репликация события не проводилась
`scope_id`	Автоматически	UUID	Идентификатор пользовательской инфраструктуры в MaxPatrol SIEM
`siem_id`	Автоматически	UUID	Идентификатор MP SIEM Server
`site_id`	Автоматически	UUID	Идентификатор площадки
`tag`	Автоматически	String	Название модуля MP 10 Collector, получившего необработанное событие
`task_id`	Автоматически	UUID	Идентификатор задачи на сбор событий
`taxonomy_version`	Автоматически	String	Версия схемы полей событий
`tenant_id`	Автоматически	UUID	Идентификатор дочерней организации в рамках площадки
`type`	Автоматически	String	Не используется

Поле	Обязательное	Тип данных	Описание
`uuid`	Автоматически	UUID	Идентификатор события. Событие в необработанном и нормализованном виде имеет один и тот же UUID

Приложение B. Значения полей с типом данных Enum

Поля событий `action`, `object`, `status` и `subject` могут принимать значения, указанные в таблицах ниже.

Таблица 276. Значения поля action

Значение	Действие
`access`	Получение доступа
`alert`	Получение предупреждения
`allow`	Разрешение
`apply`	Применение
`assign`	Сопоставление одного объекта с другим
`backup`	Резервное копирование
`bind`	Присвоение IP-адреса, добавление политики в группу
`call`	Звонок
`check`	Проверка
`clean`	Очистка
`close`	Закрытие
`configure`	Настройка
`connect`	Соединение
`copy`	Копирование
`create`	Создание
`decrypt`	Расшифровка
`deescalate`	Завершение сессии с повышенными привилегиями
`deliver`	Доставка
`deny`	Запрет
`detect`	Обнаружение
`disable`	Выключение
`disconnect`	Разъединение
`down`	Изменение состояния интерфейса на down (выключение)
`download`	Скачивание
`enable`	Включение

Значение	Действие
encrypt	Зашифровка
escalate	Начало сессии с повышенными привилегиями
exclude	Исключение
execute	Выполнение
extract	Извлечение
grant	Предоставление права, привилегии
info	События, носящие информационный характер, без возможности выделить действия
initiate	Инициирование
inject	Внедрение внешнего содержимого
install	Установка
leak	Утечка данных
lock	Блокирование
login	Вход в систему
logout	Выход из системы
modify	Изменение
move	Перемещение
open	Открывание
print	Распечатывание
protect	Защита
quarantine	Изолирование
read	Чтение или загрузка содержимого
receive	Принятие
reject	Отмена
remove	Удаление
rename	Переименование
request	Выполнение запроса на предоставление объекта или информации об объекте
reset	Сброс к состоянию по умолчанию
restart	Перезапуск
restore	Восстановление
revoke	Отзыв права, привилегии

Значение	Действие
rollback	Откат к предыдущему состоянию
scan	Сканирование
search	Поиск
send	Отправка
start	Запуск
state	Изменение состояния (на другое из заранее заданного перечня)
stop	Остановка
sync	Синхронизация
terminate	Аварийное завершение
truncate	Удаление содержимого объекта (с сохранением объекта)
unassign	Отмена сопоставления одного объекта с другим
unbind	Удаление политики из группы, освобождение IP-адреса
uninstall	Удаление продукта
unlock	Разблокирование
up	Изменение состояния интерфейса на up (включение)
update	Обновление (минорное)
upgrade	Обновление (мажорное)
upload	Загрузка
validate	Валидация
view	Просмотр

Таблица 277. Значения поля object

Значение	Описание
account	Учетная запись
alert	Оповещение
application	Приложение
arp_table	Таблица ARP
attack	Атака
certificate	Сертификат
check	Проверка
client	Клиент

Значение	Описание
cmdlet	Командлет
command	Команда
compliance	Соответствие требованиям безопасности
computer	Компьютер
configuration	Конфигурация
connection	Соединение
database	База данных
db_object	Объект базы данных
device	Устройство
ds_object	Объект службы каталогов
file	Файл
file_object	Объект файловой системы
flow	Поток данных
handle	Ссылка на ресурс в памяти процесса (файловый дескриптор)
host	Узел, для которого известен только IP-адрес или MAC-адрес
infected_object	Инфицированный объект
interface	Интерфейс
ip_address	IP-адрес
link	Ссылка
log	Файл журнала записи событий
logging	Журналирование
mailbox	Почтовый ящик
malware	Вредоносный объект
message	Сообщение
mode	Режим
module	Модуль
network	Сеть
node	Сетевой узел
package	Дистрибутив приложения
packet	Пакет
policy	Политика

Значение	Описание
port	Порт
process	Процесс
profile	Профиль
reg_object	Объект системного реестра
report	Отчет
request	Запрос
resource	Ресурс
rule	Правило
scan	Результат сканирования
service	Сервис
session	Сессия
socket	Программный интерфейс для передачи данных
system	Система
table	Таблица
task	Задача
thread	Поток команд
transaction	Транзакция
translation	Трансляция сетевого адреса (запись в таблице NAT)
trust	Доверительное отношение (например, между доменами)
tunnel	Туннель
update	Обновление
user_group	Группа пользователей
virtual_key	Цифровой сертификат, токен или ключ аутентификации (включая файлы с идентификационной информацией и лицензией)
virtual_machine	Виртуальная машина
volume	Том
vulnerability	Уязвимость

Таблица 278. Значения поля status

Значение	Описание
failure	Неуспешная попытка
success	Успешная попытка

Значение	Описание
ongoing	Действие происходит в данный момент. Результат выполнения еще неизвестен

Таблица 279. Значения поля subject

Значение	Описание
account	Пользователь (действия выполняются от имени учетной записи)
application	Приложение
host	Узел
process	Процесс
rule	Правило
service	Служба
system	Система

Приложение Г. Правила заполнения поля event_src.category

Правила заполнения поля нормализованных событий `event_src.category` для категоризации источников событий представлены в таблице ниже.

Таблица 280. Значения поля event_src.category

Значение	Описание
`AAA`	Аутентификация, авторизация, учетные записи
`Anti-virus`	Антивирусное программное обеспечение
`Application security`	Защита приложений
`Application server`	Сервер приложений (например, SAP)
`Backup server`	Система резервного копирования и восстановления
`Certification authority`	Служба сертификации (например, Microsoft Windows CA, OpenSSL)
`Database server`	Сервер баз данных
`DHCP server`	Сервер DHCP
`Directory service`	Служба каталогов (например, Microsoft AD, Novell eDirectory, OpenLDAP)
`DLP`	DLP
`DNS server`	Сервер DNS
`DPI`	Система DPI
`File service`	Файловая служба (например, Samba)
`Firewall`	Межсетевой экран
`Honeypot`	Honeypot
`Host security`	Защита узлов
`IDS/IPS`	IDS/IPS
`Mail server`	Почтовый сервер
`Mobile`	Мобильное устройство
`Network device`	Сетевое оборудование
`Network monitoring system`	Система мониторинга сети

Значение	Описание
`Network security`	Защита сети
`Operating system`	Операционная система
`Other`	Другое
`Physical security`	Система физической защиты (например, СКУД, видеонаблюдение)
`Proxy server`	Прокси-сервер
`SCADA`	Система SCADA
`SIEM`	Система SIEM
`Storage device`	Система хранения данных
`Telecom`	Система связи
`Terminal services`	Служба каталогов (например, Microsoft AD, Novell eDirectory, OpenLDAP)
`Virtualization`	Система виртуализации
`Voice over IP`	IP-телефония
`VPN`	VPN
`Web security`	Сетевая защита
`Web server`	Веб-сервер
`Wireless`	Беспроводная сеть

Приложение Д. Правила заполнения полей category.generic, category.high, category.low

Правила заполнения полей корреляционных событий для категоризации событий представлены в таблице ниже.

Таблица 281. Правила заполнения полей category.generic, category.high, category.low

Generic	High	Low
Access	Authentication	Default Credentials
		Host
		Local
		Remote
		Service
		Unknown Type
	Authorization	Host
		Network
		Object
		User
Accounting	Network Accounting	Address Translation
		Connections & Sessions
		Firewall Rules Usage
		Traffic
	User Accounting	Administrative Privilege Use
		Application & Component Launching
		Command Execution
		Data Usage
		System Accounts Use
		System Application Launching
Anomaly	Network Anomaly	Detection
	System Anomaly	
	User Behavior Anomaly	

Generic	High	Low
Asset & Asset Group Management	Asset Group Management	Activation
		Creation
		Deactivation
		Deletion
		Modification
	Asset Management	Domain Activation
		Domain Creation
		Domain Data Collection
		Domain Deactivation
		Domain Deletion
		Host Activation
		Host Creation
		Host Data Collection
		Host Deactivation
		Host Deletion
		Hypervisor Activation
		Hypervisor Creation
		Hypervisor Data Collection
		Hypervisor Deactivation
		Hypervisor Deletion
		Link Activation
		Link Creation
		Link Data Collection
		Link Deactivation
		Link Deletion
		Network Activation
		Network Creation
		Network Data Collection
		Network Deactivation
		Network Deletion
		Virtual Host Activation

Generic	High	Low
		Virtual Host Creation
		Virtual Host Data Collection
		Virtual Host Deactivation
		Virtual Host Deletion
Attacks & Recon	Attack	Bruteforce
		Complex Attack
		DDoS
		DoS
		HIPS Alert
		Identity Theft
		IDS/IPS Alert
		Miscellaneous
		Network Attack
		Post Compromise
		Potential Attack
		Privilege Escalation
		Spam Attack
		Vulnerability Exploitation
	Recon	Crawling/Dictionary Bruteforce
		Enumeration
		Fingerprinting
		Network Scan
		OS Discovery
		Port Discovery
		Service Discovery
Business Continuity	Backup/Restore	Backup Jobs
		Backup Schedule
		Restore Jobs
	Cluster Nodes	Connection
		Creation
		Deletion

Generic	High	Low
		Replication
	Replication	Data
		Heartbeat
		Settings
Compliance	Policy Violation	Application Policy Violation
		AV Policy Violation
		Confidentiality Policy Violation
		Database Policy Violation
		IM Policy Violation
		IP Access Policy Violation
		Mail Policy Violation
		P2P Policy Violation
		Remote Access Policy Violation
		System Update Policy Violation
		Web Policy Violation
Configuration Management	Network Configuration	Access Rule Modification
		Interface Modification
		Link Modification
		Port Detection
		Port State Change
		Routing Changes
	Rights Management	Critical Privileges
		Group Assignment
		Group Creation
		Group Deletion
		Group Modification
		Object Rights Modification
		Rights Assignment
		Rights Revocation
		Role Assignment
		Role Creation

Generic	High	Low
		Role Deletion
		Role Modification
		Role Revocation
		User Membership Modification
	System Configuration Management	Application Object Modification
		Application Settings Modification
		Network Service Detection
		Network Service Modification
		Network Service Resumption
		Network Service Stoppage
		OS Service Detection
		OS Service Modification
		OS Service Resumption
		OS Service Stoppage
		OS Settings Modification
		Security Settings Modification
	User Management	Creation
		Deletion
		Detection
		External User Connection
		Locking
		Modification
		Unlocking
Hacktool	Bruteforcer	Deletion
	DoS	Detection
	Encryption	Exploitation
	Enumeration	Installation
	Exploit Kit	
	Fingerprint	
	Miscellaneous	
	MITM	

Generic	High	Low
	Remote Shell	
	Scanner	
	Shellcode	
	Sniffer	
	Spoofing	
	Tunnel	
	Web Scanner	
Hardware Management	CPU Management	CPU Detection
		CPU Resources Change
	External Device Management	Bluetooth pairing
		Mobile device
		Printer
		Scanner
		USB Device
	External Storage Management	External Disk Detection
		External Disk Mounting
		External Disk Unmounting
		SAN/NAS Detection
		SAN/NAS Mounting
		SAN/NAS Unmounting
	Hardware inventory	Hardware Configuration Errors
		Hardware Detection
		Hardware Installation
		Hardware Removal
	Memory Management	Memory Resizing
	Storage Management	Local Storage
		Network Storage
Incident Management	Incident	Creation
		Deletion
		Detection
		Modification

Generic	High	Low
		Remediation
Information Management	Data Loss	Accidental
		Deliberate
	Data Modification	By System
		By User
	Information Access Control	Assignment
		Object Access
		Settings
	Information Labeling	Label Assignment
		Label Management
	Information Leak	Confidential Information
		Critical Information
	Leakage Channel	External Drive
		Mail
		Mobile Device
		Printer
		Web
Malware	Backdoor	Curing
	Bootkit	Detection
	Botnet	Epidemic
	Miscellaneous	Mitigation
	Rootkit	
	Trojan	
	Virus	
	Worm	
Monitoring	Errors	Application
		Hardware
		Network
		System
	Network Monitoring	Interface State
		Performance

Generic	High	Low
	System Monitoring	Notification
		Performance
		Processes
		System State
Network Interaction	Network Interaction	Application
		Database
		File transfer
		Mail transfer
		Remote Management
		Suspicious
		Unknown
		VPN
		Web
Software Management	Operating System Management	Deletion
		Detection
		Installation
		Update
	Patch Management	Deletion
		Installation
	Security Software Management	Deletion
		Detection
		Installation
		Update
	Software Inventory	Deletion
		Detection
		Installation
		Update
Vulnerability Management	Vulnerability	Knowledge Base Updates
		Potential Exploit Detection
		Vulnerability Detection
		Vulnerability Exception

Generic	High	Low
		Vulnerability Remediation

Приложение Е. Правила заполнения полей инцидента category и type

Для инцидента могут быть указаны категория в поле `category` и тип в поле `type`. Для инцидентов категории «Атака» также могут быть указаны параметры.

Таблица 282. Правила заполнения полей category, type и полей параметров инцидента

category	Категория	type	Тип	Поля для параметров	
`Attack`	Атака	`BotNetCC`	Командный центр бот-сети	`BotNetInfo`	Общие сведения
				`BotNetServers`	Серверы C&C
				`BotNetType`	Тип бот-сети
		`BotNetNode`	Узел бот-сети	`BotNetInfo`	Общие сведения
				`BotNetServers`	Серверы C&C
				`BotNetType`	Тип бот-сети
		`BruteForce`	Подбор данных	`BytesPerSecond`	Мощность атаки — байт в секунду
				`PacketsPerSecond`	Мощность атаки — пакетов в секунду
				`ProtocolType`	Тип протокола
		`DoSDDoS`	Отказ в обслуживании	`AttackType`	Тип атаки
				`BytesPerSecond`	Мощность атаки — байт в секунду

category	Категория	type	Тип	Поля для параметров	
				PacketsPerSecond	Мощность атаки — пакетов в секунду
		MaliciousResources	Вредоносные ресурсы	MaliciousHashSum	Хеш-суммы файлов
				MaliciousResource	Вредоносные ресурсы
				MaliciousResourceType	Тип вредоносного ресурса
				MaliciousSoftwareIds	Идентификаторы вредоносного ПО
				MaliciousSoftwareType	Тип вредоносного ПО
		NetworkScan	Сканирование сети	ScanMethods	Методы сканирования
				ScanPorts	Список сканируемых портов
		Phishing	Фишинг	LegitimateResource	Легитимные ресурсы
				MaliciousResources	Вредоносные ресурсы
		Spam	Спам	SpamMailCount	Количество почтовых сообщений

category	Категория	type	Тип	Поля для параметров	
		UnauthorizedAccess	НСД (несанкционированный доступ)	UnauthorizedAccessEffect	Последствия НСД
				UnauthorizedAccessProtocol	Протокол
				UnauthorizedAccessType	Способ получения НСД
		VulnerabilityExploitation	Эксплуатация уязвимости	VulnerabilityExploitationEffect	Последствия эксплуатации уязвимости
				VulnerabilityExploitationType	Класс уязвимости
DataSecurity	Безопасность данных	DataLeakage	Утечка данных	—	
		UnauthorizedDataChange	Несанкционированное изменение данных	—	
		UnauthorizedDataDeletion	Несанкционированное удаление данных	—	
Malfunction	Нарушение работоспособности	BackupMalfunction	Нарушение работоспособности при резервном копировании данных	—	
		DOSAttack	Атака типа «Отказ в обслуживании»	—	
		ProtectionMalfunction	Неработоспособность средств защиты	—	
		ServiceMalfunction	Нарушение работоспособности сервиса	—	

category	Категория	type	Тип	Поля для параметров
		SoftwareMalfunction	Нарушение работоспособности ПО	—
MalwareDetection	Обнаружение вредоносного ПО	HackToolsDetection	Обнаружение хакерского ПО	—
		TrojanHorseDetection	Обнаружение троянской программы	—
		VirusDetection	Обнаружение вируса	—
		WormDetection	Обнаружение сетевого червя	—
PolicyViolation	Нарушение политик ИБ	ForbiddenServiceUsage	Использование запрещенных сервисов	—
		PatchPolicyViolation	Нарушение регламента по установке обновлений	—
		PermissionViolation	Превышение служебных полномочий	—
		SoftwareInstallationPolicyViolation	Установка запрещенного ПО	—
UnauthorizedAccess	Неавторизованный доступ	HostOrSoftCompromising	Компрометация узла/ПО	—
		ProtectionBypassing	Обход средств защиты	—

category	Категория	type	Тип	Поля для параметров
		SecurityPerimeterBreach	Нарушение границ внешнего периметра	—
		UserCompromising	Компрометация пользователя	—
Undefined	Не определена	SoftwareSuspiciousActivity	Подозрительные операции с ПО	—
		Undefined	Не определен	—
VulnerabilityManagement	Управление уязвимостями	CriticalVulnerabilityDetection	Обнаружение критически опасной уязвимости	—
		SecurityBackdoorDetection	Обнаружение бэкдора	—

Предметный указатель

Символы

A

B

C

D

E

F

H

I

J

K

L

M

N

O

Q

R

S

T

U

W

X

Y

А

И

К

М

Н

О

П

С

Т

Ф

positive
technologies

Positive Technologies — лидер рынка результативной кибербезопасности. Компания является ведущим разработчиком продуктов, решений и сервисов, позволяющих выявлять и предотвращать кибератаки до того, как они причинят неприемлемый ущерб бизнесу и целым отраслям экономики. Наши технологии используют более 3300 организаций по всему миру, в том числе 80% компаний из рейтинга «Эксперт-400». Positive Technologies — первая и единственная компания из сферы кибербезопасности на Московской бирже (MOEX: POSI), у нее более 170 тысяч акционеров.