3 Низкоуровневое обнаружение

Обзор

Низкоуровневое обнаружение (LLD) даёт возможность автоматического создания элементов данных, триггеров и графиков для различных объектов на компьютере. Например, Zabbix может автоматически начать мониторить файловые системы или сетевые интерфейсы с вашего устройства, без необходимости создания вручную элементов данных для каждой файловой системы или сетевого интерфейса. Кроме того, в Zabbix имеется возможность настроить удаление ненужных объектов, основываясь на фактических результатах периодически выполняемого обнаружения.

Пользователь имеет возможность определить свои собственные типы обнаружения, обеспечив их функционирование согласно спецификации JSON протокола.

Общая архитектура процессов обнаружения заключается в следующем.

Сначала, пользователь создает правило обнаружения в “Настройка” → “Шаблоны” → колонка “Обнаружение”. Правило обнаружения состоит из (1) элемента данных, который осуществляет обнаружение необходимых объектов (например, файловые системы или сетевые интерфейсы) и (2) прототипов элементов данных, триггеров и графиков, которые должны быть созданы на основании полученных значений этого элемента данных.

Элемент данных, который осуществляет обнаружение необходимых объектов, подобен обычным элементам данных, которые видны в других местах: Zabbix сервер запрашивает у Zabbix агента (или любой другой указанный тип элемента данных) значение этого элемента данных, и агент отвечает текстовым значением. Разница в том, что значение, которое возвращает агент, должно содержать список обнаруженных объектов в специальном JSON формате. Хотя детали этого формата важны только для создателей собственных проверок обнаружения, всё же всем необходимо знать, что возвращаемое значение содержит список из пар: макрос → значение. Например, элемент данных “net.if.discovery” может вернуть две пары: “{#IFNAME}” → “lo” и “{#IFNAME}” → “eth0”.

Эти макросы затем используются в именах, ключах и в других полях прототипов, которые являются основой для создания реальных элементов данных, триггеров и графиков каждому обнаруженному объекту. Смотрите полный список опций по использованию макросов в низкоуровневом обнаружении.

Когда сервер получает значение элемента данных обнаружения, он смотрит на пару макрос → значение и для каждой пары создает реальные элементы данных, триггеров и графиков, основанных на их прототипах. В приведенном выше примере с “net.if.discovery”, сервер будет создавать один набор элементов данных, триггеров и графиков для локального интерфейса “lo” и другой набор для интерфейса “eth0”.

Обратите внимание, что начиная с **Zabbix 4.2**, формат JSON, возвращаемый правилами низкоуровневого обнаружения, был изменен. Больше не ожидается, что JSON будет содержать объект "data". Низкоуровневое обнаружение теперь будет принимать обычный JSON, содержащий массив, чтобы поддерживать новые функции, такие как предобработка значения элемента данных и пользовательские пути к макросам низкоуровневого обнаружения в документе JSON.

Встроенные ключи обнаружения были обновлены, чтобы возвращать массив строк LLD в корне документа JSON. Zabbix автоматически извлечет макрос и значение, если поле массива использует синтаксис {#MACRO} в качестве ключа. Любые новые нативные проверки обнаружения будут использовать новый синтаксис без элементов "data". При обработке значения низкоуровневого обнаружения сначала определяется корень (массив в «$.» Или «$.data»).

Хотя элемент "data" был удален из всех родных элементов данных, связанных с обнаружением, для обеспечения обратной совместимости Zabbix по-прежнему будет принимать нотацию JSON с элементом "data", хотя его использование не рекомендуется. Если JSON содержит объект только с одним элементом массива "data", то он автоматически извлечет содержимое элемента, используя JSONPath `$.data`. Низкоуровневое обнаружение теперь принимает необязательные пользовательские макросы LLD с пользовательским путем, указанным в синтаксисе JSONPath.

В результате вышеуказанных изменений более новые агенты больше не смогут работать со старым Zabbix-сервером.

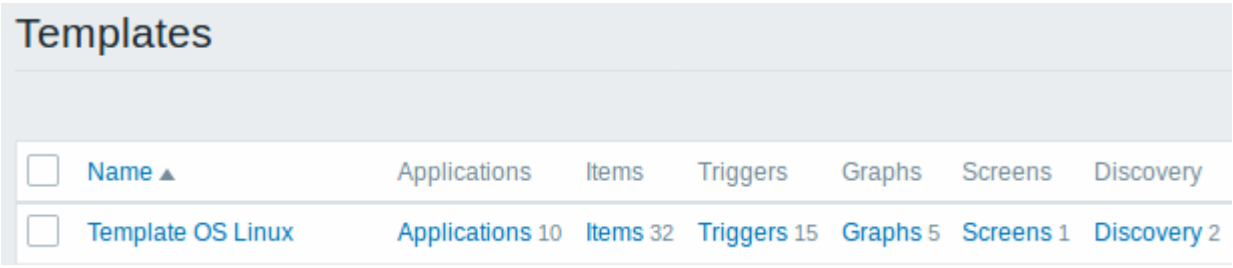
См. также: Обнаруженные объекты

Настройка низкоуровневого обнаружения

Мы проиллюстрируем низкоуровневое обнаружение на примере обнаружения файловых систем.

Для настройки обнаружения, выполните следующее:

- Перейдите в: *Настройка* → *Шаблоны*
- Нажмите на *Обнаружение* в строке с соответствующим шаблоном



- Нажмите на *Создать правило обнаружения* в верхнем правом углу экрана
- Заполните диалог правила обнаружения необходимыми деталями

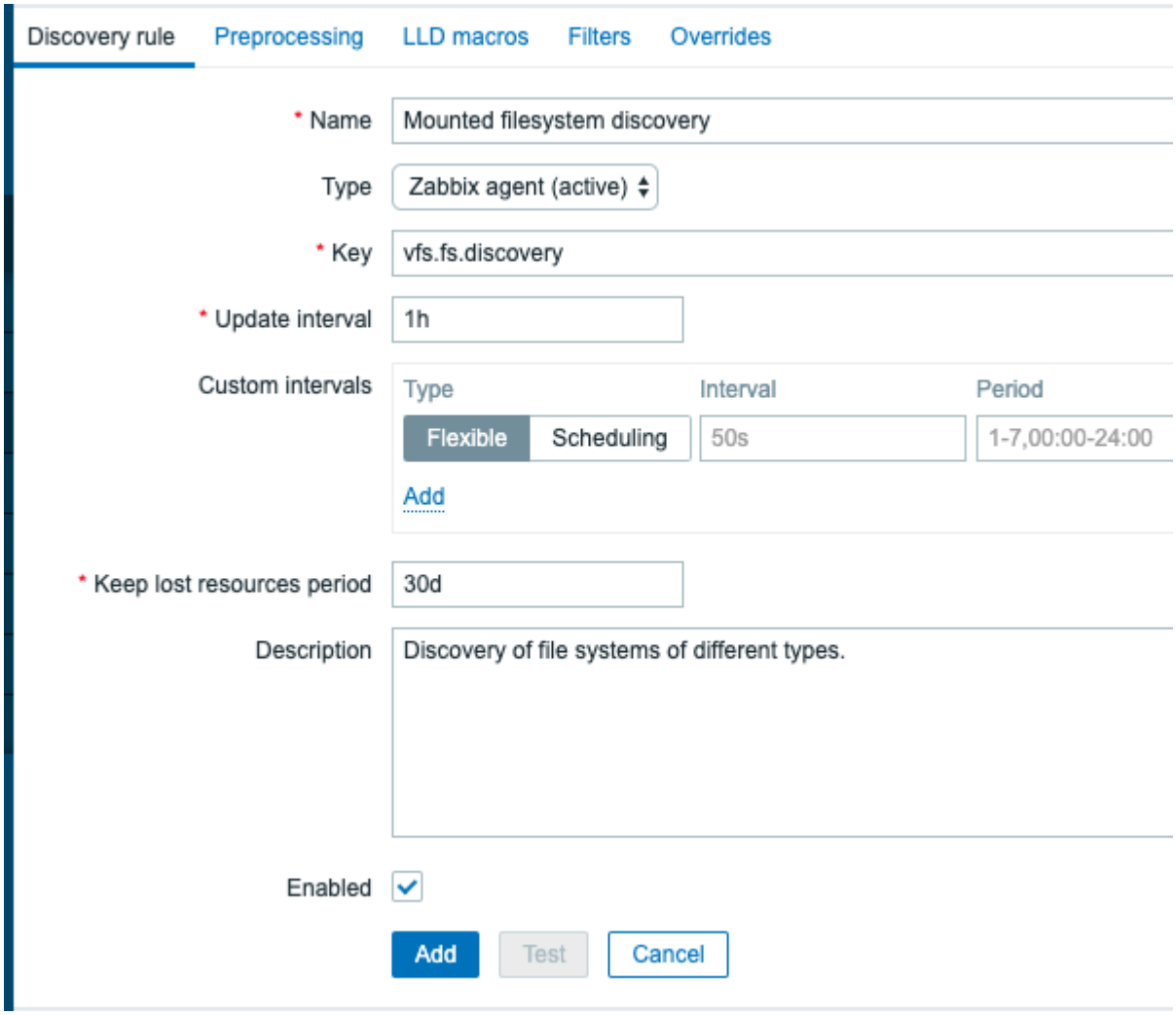
Правило обнаружения

Форма правила обнаружения содержит пять вкладок, представляющих, слева направо, поток данных во время обнаружения:

- Правило обнаружения - указывает, что наиболее важно: встроенный элемент или пользовательский сценарий для извлечения данных обнаружения

- Предварительная обработка - применяет некоторую предварительную обработку к обнаруженным данным
- Макросы LLD - позволяют извлечь некоторые значения макросов для использования в обнаруженных элементах, триггерах и т. д.
- Фильтры - позволяет фильтровать обнаруженные значения
- Переопределения - позволяет изменять элементы, триггеры, графики или прототипы хоста при применении к определенным обнаруженным объектам.

Вкладка **Правило обнаружения** содержит общие атрибуты правила обнаружения:



Все обязательные поля ввода отмечены красной звёздочкой.

Параметр	**Описание**
Имя	Имя правила обнаружения.
Тип	Тип проверки выполняемого обнаружения. В этом примере мы используем ключ элемента данных Zabbix агента. Правило обнаружения также может быть зависимым элементом данных, зависящим от обычного элемента данных (но не от другого правила обнаружения). Для зависимого элемента данных выберите соответствующий тип (Зависимый элемент данных) и укажите основной элемент в поле «Основной элемент даных». Основной элемент должен существовать.

Параметр	Описание
Ключ	Введите ключ элемента обнаружения (до 2048 символов). Например, вы можете использовать встроенный ключ элемента «vfs.fs.discovery» для возврата JSON со списком файловых систем, имеющихся на компьютере, и их типами. Обратите внимание, что другой опцией для обнаружения файловой системы является использование результатов обнаружения ключом агента «vfs.fs.get», поддерживаемым начиная с Zabbix 4.4.5 (см. пример)
Интервал обновления	Это поле задает как часто Zabbix выполняет обнаружение. В начале, когда вы только настраиваете обнаружение файловых систем, вы можете указать маленький интервал, но как только вы удостоверитесь что всё работает, вы можете установить его в 30 минут или более, потому что обычно файловые системы не меняются очень часто. Поддерживаются суффиксы времени, например 30s, 1m, 2h, 1d, начиная с Zabbix 3.4.0. Пользовательские макросы поддерживаются начиная с Zabbix 3.4.0. *Обратите внимание*: Если укажите значение равное '0', элемент данных не будет обрабатываться. Однако, если также существует переменный интервал с ненулевым значением, элемент данных будет обрабатываться в течении действия переменного интервала. *Обратите внимание*, что уже созданное правило обнаружение можно выполнить незамедлительно нажатием кнопки *Проверить сейчас* .
Пользовательские интервалы	Вы можете создавать пользовательские правила проверки элемента данных: **Гибкий** - создание исключений из Интервала обновления (интервал с другой частотой обновления) **По расписанию** - создание пользовательского расписания проверки. Для получения более подробной информации смотрите Пользовательские интервалы. Проверка по расписанию поддерживается начиная с Zabix 3.0.0.
Период сохранения потерянных ресурсов	Это поле позволяет вам указать как много дней обнаруженный объект будет храниться (не будет удален), как только его состояние обнаружения станет “Не обнаруживается более” (мин 1 час, макс 25 лет). Поддерживаются суффиксы времени, например 30s, 1m, 2h, 1d, начиная с Zabbix 3.4.0. Пользовательские макросы поддерживаются начиная с Zabbix 3.4.0. *Обратите внимание:* Если значение равно “0”, объекты будут удалены сразу. Использование значения “0” не рекомендуется, так как простое ошибочное изменение фильтра может закончится тем, что объект будет удален вместе со всеми данными истории.
Описание	Введите описание.
Состояние	Если отмечено, правило будет обрабатываться.

История правил обнаружения не сохраняется.

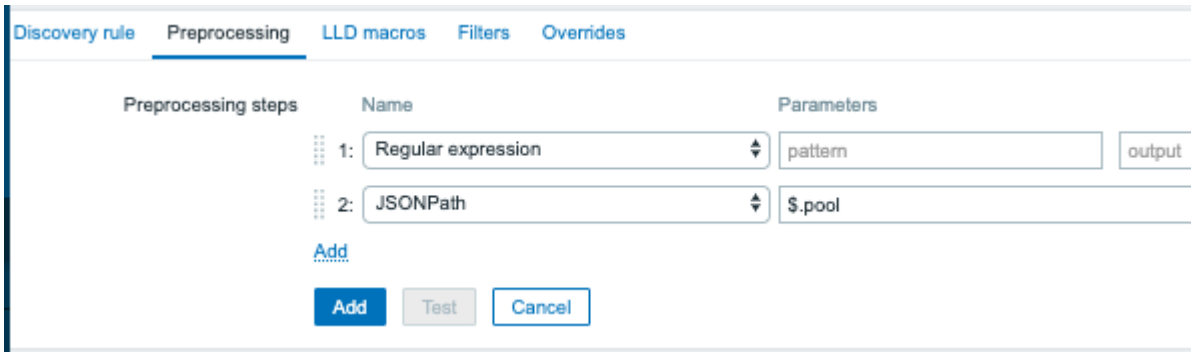
Предобработка

Вкладка **Предобработка** позволяет определить правила преобразования, применяемые к результату обнаружения. На этом этапе возможно одно или несколько преобразований. Преобразования выполняются в том порядке, в котором они определены. Вся предварительная

обработка выполняется Zabbix сервером.

См. также:

- Детали предобработки
- Тестирование предобработки



Тип	Преобразование	Описание
Текст		
	Регулярное выражение	Полученное значение сопоставляется с регулярным выражением <pattern> и заменяется извлеченным <output>. Регулярное выражение поддерживает извлечение максимум 10 групп захвата с помощью последовательности \N. Параметры: **шаблон** - регулярное выражение **вывод** - шаблон форматирования вывода. Управляющая последовательность \N (где N = 1... 9) заменяется N-нной совпадающей группой. Экранирующая последовательность \0 заменяется соответствующим текстом. Если вы установите флажок *Другое при ошибке*, появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо установить указанное сообщение об ошибке.
	Замена	Найти строку поиска и заменить ее другой (или пустотой). Все совпадающие строки поиска будут заменены. Параметры: **строка поиска** - строка для поиска и замены, с учетом регистра (обязательно) **замена** - строка для замены строки поиска. Строка замены также может быть пустой, что позволяет эффективно удалять строку поиска при ее обнаружении. Можно использовать управляющие последовательности для поиска или замены разрывов строк, возврата каретки, табуляции и пробелов "\ n \ r \ t \ s "; обратную косую черту можно экранировать как "\\", а управляющие последовательности можно экранировать как "\\n" . Экранирование разрывов строк, возврата каретки, вкладок выполняется автоматически при низкоуровневом обнаружении. Поддерживается с 5.0.0.
Составные данные		

Тип	Преобразование	Описание
	JSONPath	Будет извлечено значение или фрагмент из данных JSON с помощью функциональности JSONPath. Если вы установите флажок *Другое при ошибке*, элемент не станет неподдерживаемым в случае ошибки этапа предобработки, вместо этого появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо задать указанное сообщение об ошибке.

	XML XPath	Извлечение значения или фрагмента из XML-данных с использованием функций XPath.
		Для работы этой опции Zabbix сервер должен быть скомпилирован с поддержкой libxml.
		Примеры:
		`number(/document/item/value)` извлечет `10` из `<document><item><value>10</value></item></document>`
		`number(/document/item/@attribute)` извлечет `10` из `<document><item attribute="10"></item></document>`
		`/document/item`извлечет `<item><value>10</value></item>` из `<document><item><value>10</value></item></document>`
		Обратите внимание, что пространства имен не поддерживаются.
		Поддерживается начиная с 4.4.0.
		Если вы установите флажок *Другое при ошибке*, появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо задать указанное сообщение об ошибке.

	CSV в JSON	Данные файла CSV конвертируются в формат JSON. Для получения дополнительной информации см. : Преобразование CSV в JSON. Поддерживается с 4.4.0.
Пользовательские скрипты		
	JavaScript	Введите код JavaScript в блок, который появляется при нажатии в поле параметра или на иконку карандаша. Обратите внимание, что доступная длина JavaScript зависит от используемой базы данных. Для получения дополнительной информации см. : Предобработка Javascript (страница доступна только на английском языке)
Валидация		

Тип	Преобразование	Описание
	Не совпадает с регулярным выражением	Укажите регулярное выражение, которому значение не должно соответствовать. Например. `Error:(.*?)\.` Если вы установите флажок *Другое при ошибке*, этого появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо задать указанное сообщение об ошибке.
	Проверьте на ошибку в JSON	Проверяется, нет ли сообщения об ошибке на уровне приложения в JSONpath. Обработка будет остановлена в случае положительного результата (сообщение присутствует); в противном случае обработка будет продолжена со значением, подготовленным до этого этапа предварительной обработки. Обратите внимание, что эти внешние сервисные ошибки сообщаются пользователю напрямую, без добавления информации о шаге предварительной обработки. Например, `$.errors`. Если получен JSON-подобный `{"errors":"e1"}`, следующий шаг предобработки не будет выполнен. Если вы установите флажок *Другое при ошибке*, этого появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо задать указанное сообщение об ошибке.
	Проверьте на наличие ошибок в XML	Проверяется, нет ли сообщения об ошибке на уровне приложения в JSONpath. Обработка будет остановлена в случае положительного результата (сообщение присутствует); в противном случае обработка будет продолжена со значением, подготовленным до этого этапа предварительной обработки. Обратите внимание, что эти внешние сервисные ошибки сообщаются пользователю напрямую, без добавления информации о шаге предварительной обработки. Например, `$.errors`. Если получен JSON-подобный `{"errors":"e1"}`, следующий шаг предобработки не будет выполнен. Если вы установите флажок *Другое при ошибке*, этого появится возможность указать пользовательские параметры обработки ошибок: либо сбросить значение, либо задать указанное значение, либо задать указанное сообщение об ошибке.
Троттлинг		
	Отбрасывать не изменившееся с периодическим контролем	Отбросить зачение, если оно не изменилось в течние определенного периода (в секундах). Поддерживаются положительные целые значения для указания секунд (минимум - 1 секунда). В этом поле можно использовать суффиксы времени (например, 30 с, 1 м, 2 ч, 1 д). В этом поле можно использовать пользовательские макросы и макросы низкоуровневого обнаружения. Для элемента данных обнаружения можно указать только один параметр троттлинга. Например, `1m`. Если идентичный текст будет передан в это правило дважды в течение 60 секунд, он будет отброшен. *Примечание*: Изменение прототипов элементов данных не приводит к сбросу троттлинга. Троттлинг сбрасывается только при изменении шагов предварительной обработки.
Prometheus		
	Prometheus в JSON	Преобразуйте необходимые метрики Prometheus в JSON. См. Проверки Prometheus (страница доступна только на английском языке) для получения более подробной информации.

Обратите внимание, что если правило обнаружения было применено к хосту через шаблон, то

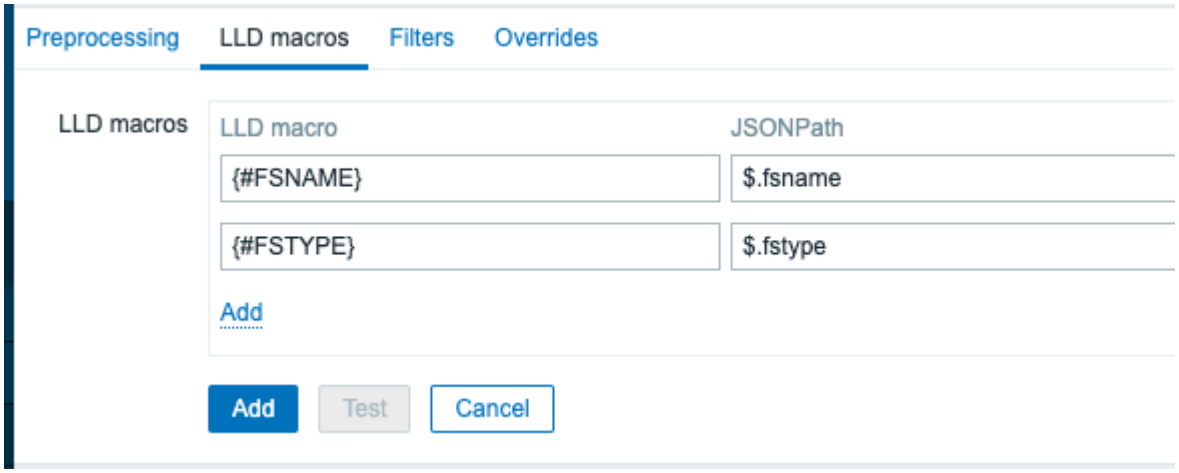
содержимое этой вкладки доступно только для чтения.

Пользовательские макросы

На вкладке **LLD макросы** можно указать пользовательские макросы низкоуровневого обнаружения.

Пользовательские макросы полезны в тех случаях, когда возвращаемый JSON не имеет уже определенных макросов, которые необходимы. Так, например:

- Родной ключ `vfs.fs.discovery` для обнаружения файловой системы возвращает JSON с некоторыми предопределенными макросами LLD, такими как {#FSNAME}, {#FSTYPE}. Эти макросы можно использовать непосредственно в элементах данных, прототипах триггеров (см. следующие разделы страницы); в этом случае задавать пользовательские макросы не требуется;
- Элемент данных агента `vfs.fs.get` также возвращает JSON с данными файловой системы, но без каких-либо предопределенных макросов LLD. В этом случае вы можете определить макросы самостоятельно и сопоставить их со значениями в JSON, используя JSONPath:



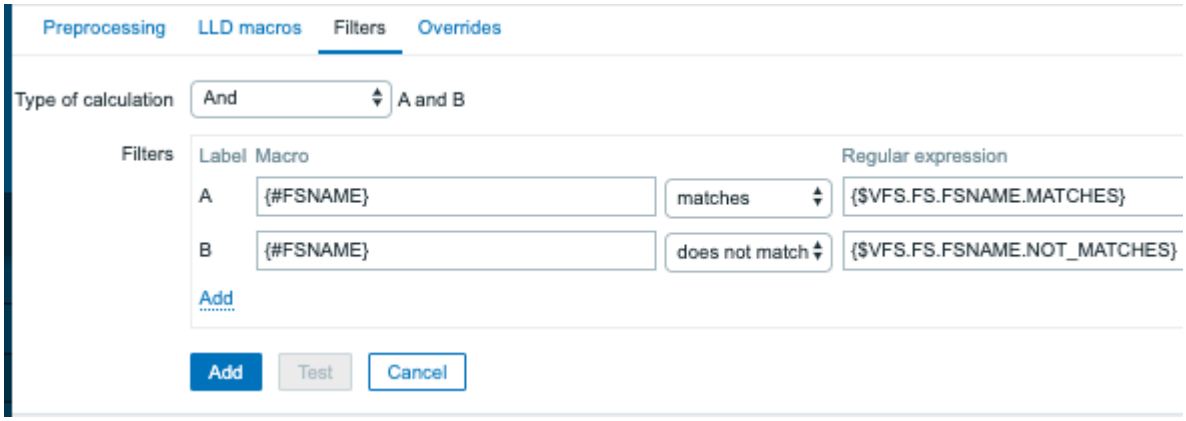
Извлеченные значения могут использоваться в обнаруженных элементах, триггерах и т. Д. Обратите внимание, что значения будут извлечены из результата обнаружения и любых шагов предобработки, выполненных до этого момента.

Параметр	Описание
Макрос LLD	Имя макроса низкоуровневого обнаружения, используя следующий синтаксис: {#MACRO}.

Параметр	Описание
JSONPath	Путь, используемый для извлечения значения макроса LLD из строки LLD с использованием синтаксиса JSONPath. Например, `$.foo` будет извлекать "bar" и "baz" из этого JSON: `[{"foo":"bar"}, {"foo":"baz"}]` Значения, извлеченные из возвращенного JSON, используются для замены макросов LLD в полях прототипов элементов данных, триггеров, и т. д. JSONPath можно указать с помощью точечной или скобочной нотации. Обозначение в скобках следует использовать в случае любых специальных символов и Unicode, например `$['unicode + special chars #1']['unicode + special chars #2']`.

Фильтр правила обнаружения

Вкладка **Фильтры** содержит определения фильтрации правила обнаружения:



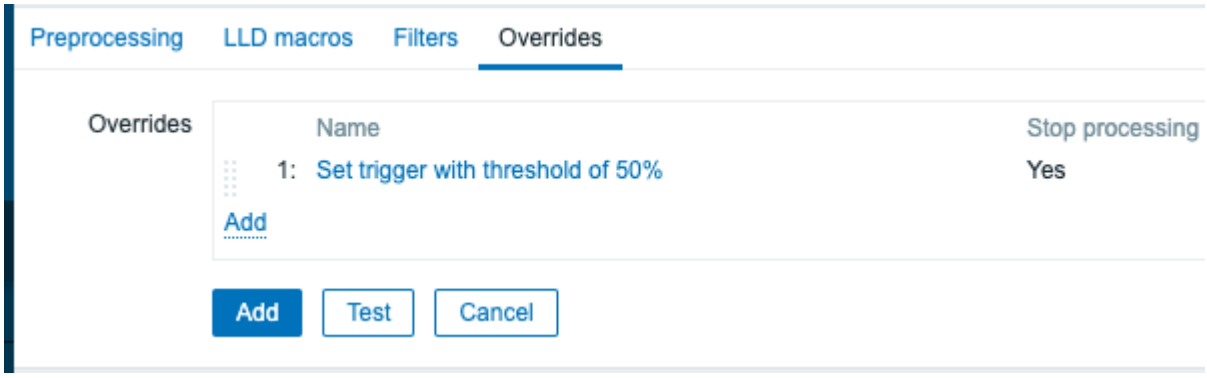
Параметр	Описание
Тип вычисления	Доступны следующие опции расчета фильтров: **И** - должны выполниться все фильтры; **Или** - достаточно выполнения одного фильтра; **И/Или** - используется *И* для разных имен макросов и *Или* с одинаковым именем макроса; **Пользовательское выражение** - появляется возможность указать пользовательское вычисление фильтров. Формула должна включать в себя все фильтры из списка. Ограничено 255 символами.

Параметр	Описание
Фильтры	Фильтр можно использовать только для генерирования реальных элементов данных, триггеров и графиков конкретных файловых систем. Ожидается использование Perl Compatible Regular Expression (PCRE). Например, если вы заинтересованы только в файловых системах C:, D: и E:, вы можете поместить {#FSNAME} в поле “Макрос” и регулярное выражение "^C|^D|^E" в текстовые поля “Регулярное выражение”. Фильтрация также возможна по типам файловых систем, при использовании макроса {#FSTYPE} (например, "^ext|^reiserfs") и по типу диска (поддерживается только Windows агентов), используя макрос {#FSDRIVETYPE} (например, "fixed"). Вы можете ввести в поле “Регулярное выражение” регулярное выражение или ссылку на глобальное регулярное выражение. Для проверки регулярного выражения вы можете использовать “grep -E”, например: `for f in ext2 nfs reiserfs smbfs; do echo $f | grep -E '^ext|^reiserfs' || echo "SKIP: $f"; done` Макрос {#FSDRIVETYPE} на Windows поддерживается начиная с Zabbix **3.0.0**. Определение нескольких фильтров поддерживается начиная с **2.4.0**. Обратите внимание, что если какой-то макрос из фильтра отсутствует в ответе, найденный объект будет проигнорирован. Две опции выпадающего меню в фильтре позволяют выбрать, должен ли макрос соответствовать указанному регулярному выражению или наоборот, не соответствовать ему.

Чтобы обнаружение сработало корректно, база данных Zabbix в MySQL должна быть создана чувствительной к регистру, если имена файловых систем различаются только по регистру.
Ошибка или опечатка в регулярном выражении, которое используется в LLD правиле, может привести к удалению тысяч элементов конфигурации, данных истории и событий на большом количестве узлов сети. Например, некорректное регулярное выражение “File systems for discovery” может привести к удалению тысяч элементов данных, триггеров, данных истории и событий.

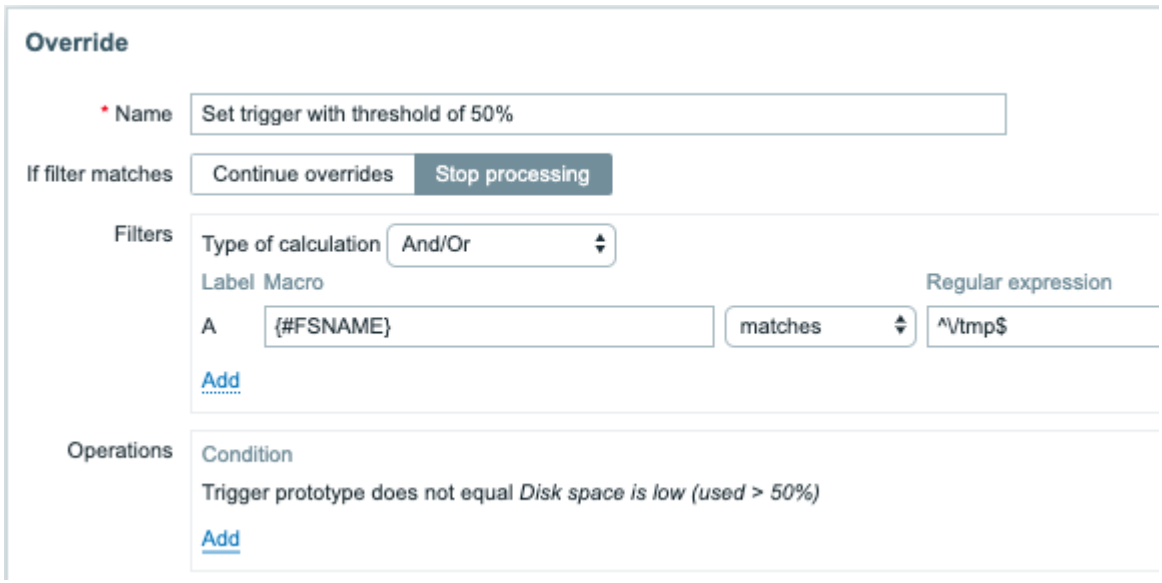
Замещения

Вкладка **Замещения** позволяет устанавливать правила для изменения списка прототипов элементов данных, триггеров, графиков и узлов сети или их атрибутов, если обнаруженный объект соответствует заданным критериям.



Замещения отображаются и выполняются в том порядке, в котором они определены. При необходимости этот порядок можно изменить, перетаскивая строчки из списка с помощью мышки. Чтобы настроить детали нового замещения, нажмите *Добавить*. Чтобы изменить

существующее замещение, нажмите на его название. Откроется всплывающее окно, позволяющее редактировать заданные настройки.

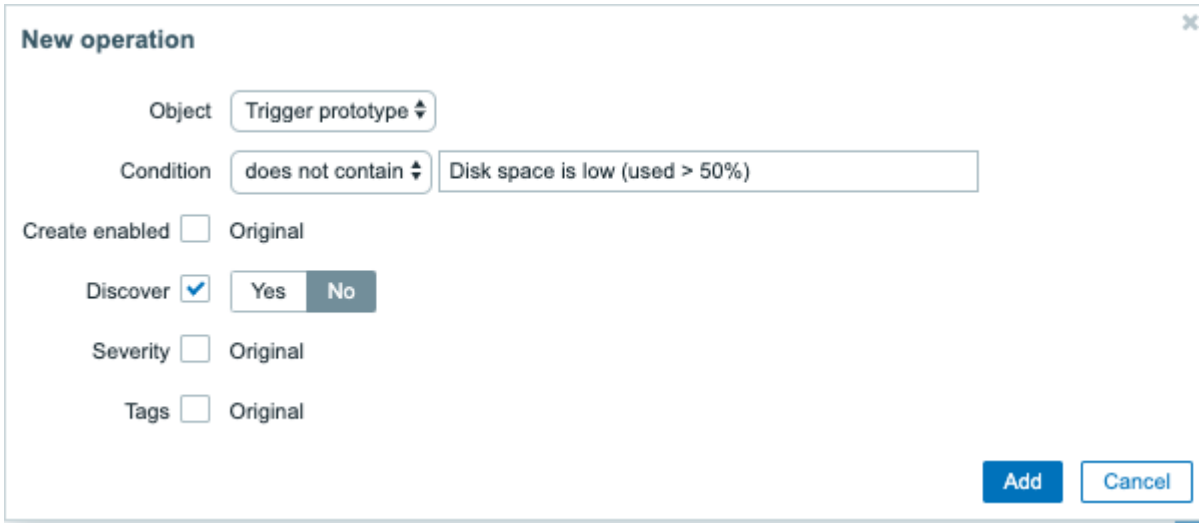


Все обязательные параметры отмечены красными звездочками.

Параметр	Описание
Имя	Уникальное (для каждого правила низкоуровневого обнаружения) имя замещения.
Если фильтр соответствует	Следует ли обрабатывать следующие замещения при выполнении условий фильтра: **Продолжить замещения** - будут обработаны последующие замещения. **Остановить обработку** - будут выполнены предшествующие операции (если есть), и текущее замещение, последующие замещения будут проигнорированы.
Фильтры	Определяет, к каким обнаруженным объектам следует применить замещение. Фильтры замещения обрабатываются после фильтров правила обнаружения и имеют ту же функциональность.
Операции	Операции замещения отображаются со следующими подробностями: **Условие** - тип объекта (прототип элемента данных/прототип триггеров/прототип графиков/прототип узлов сети) и условие, которое должно быть выполнено (равно/не равно/содержит/не содержит/совпадает/не соответствует) **Действие** - отображаются ссылки для редактирования и удаления операции..

Настройка операций

Чтобы настроить детали новой операции, нажмите Добавить в блоке операций. Чтобы редактировать существующую операцию, нажмите Изменить рядом с операцией. Откроется всплывающее окно, в котором вы можете редактировать детали операции.



Параметр		**Описание**
Объект		Доступны четыре типа объектов: Прототип элемента данных Прототип триггеров Прототип графиков Прототип узлов сети
Условие		Позволяет фильтровать объекты, к которым должна применяться операция.
	Опция	Поддерживаются следующие опции: **равно** - применить к этому прототипу **не равно** - применить ко всем прототипам, кроме этого **содержит** - имя прототипа содержит этот шаблон **не содержит** - имя прототипа не содержит этот шаблон **совпадает** - имя прототипа совпадает с этим шаблоном **не соответствует** - имя прототипа не соответствует этому шаблону
	Шаблон	Шаблон для соответствия имени элемента, триггера, графика или хоста в зависимости от выбранного объекта.

Параметр	Описание
Объект: *Прототип элемента данных*	
Создать активированным	При установке флажка появятся кнопки, позволяющие переопределить исходные параметры объекта: *Да* - объект будет добавлен в активированном состоянии *Нет* - объект будет добавлен к обнаруженному объекту, но в деактивированном состоянии.
Обнаружить	При установке флажка появятся кнопки, позволяющие переопределить настройки прототипа исходного объекта: *Да* - объект будет добавлен. *Нет* - объект не будет добавлен.
Интервал обновления	При установке флажка появятся две опции, позволяющие изменить интервал обновления для элемента данных: *Задержка* - интервал обновления элемента данных. Поддерживаются пользовательские макросы и суффиксы времени (например, 30s, 1m, 2h, 1d). Должен быть равен 0, если используются *Пользовательские интервалы*. *Пользовательские интервалы* - нажмите Добавить, чтобы указать переменный интервал или итервал по расписанию. См. также пользовательские интервалы.
Период хранения истории	При установке флажка появятся кнопки, позволяющие установить другой период хранения истории для элемента данных: *Не хранить историю* - если выбрано, данные не будут сохраняться. *Период хранения* - если выбрано, справа появится поле ввода для указания периода хранения. Поддерживаются пользовательские макросы и макросы низкоуровневого обнаружения.
Период хранения динамики изменений	При установке флажка появятся кнопки, позволяющие установить другой период хранения динамики изменений для элемента данных: *Не хранить динамику изменений* - если выбрано, данные не будут сохраняться. *Период хранения* - если выбрано, справа появится поле ввода для указания периода хранения. Поддерживаются пользовательские макросы и макросы низкоуровневого обнаружения.

Параметр	Описание
Объект: *Прототип триггеров*	
Создать активированным	При установке флажка появятся кнопки, позволяющие переопределить исходные параметры объекта: *Да* - объект будет добавлен в активированном состоянии *Нет* - объект будет добавлен к обнаруженному объекту, но в деактивированном состоянии.
Обнаружить	При установке флажка появятся кнопки, позволяющие переопределить настройки прототипа исходного объекта: *Да* - объект будет добавлен. *Нет* - объект не будет добавлен.
Важность	При установке флажка появятся кнопки уровней важности триггера, позволяющие изменить важность триггера.
Теги	При установке флажка появится новый блок, позволяющий указать пары тег-значение. Все теги прототипа триггера будут заменены тегами из этого переопределения.
Объект: *Прототип графиков*	
Обнаружить	При установке флажка появятся кнопки, позволяющие переопределить исходные настройки прототипа графиков: *Да* - график будет добавлен. *Нет* - график не будет добавлен.
Объект: *Прототип узлов сети*	
Создать активированным	При установке флажка появятся кнопки, позволяющие переопределить исходные параметры объекта: *Да* - объект будет добавлен в активированном состоянии *Нет* - объект будет добавлен к обнаруженному объекту, но в деактивированном состоянии.
Обнаружить	При установке флажка появятся кнопки, позволяющие переопределить настройки прототипа исходного объекта: *Да* - объект будет добавлен. *Нет* - объект не будет добавлен.
Присоединить шаблоны	При установке флажка появится поле ввода для указания шаблонов. Начните вводить имя шаблона или нажмите *Выбрать* рядом с полем и выберите шаблоны из списка во всплывающем окне. Все шаблоны, связанные с прототипом узлов сети, будут заменены шаблонами из этого замещения.
Инвентарные данные узла сети	При установке флажка появятся кнопки, позволяющие выбрать другой режим заполнения инвентарных данных для прототипа узла сети: *Деактивировано* - не заполнять инвентарные данные узла сети *Вручную* - возможность предоставить данные вручную *Автоматически* - автоматически заполнять данные инвентаризации узла сети на основе собранных метрик.

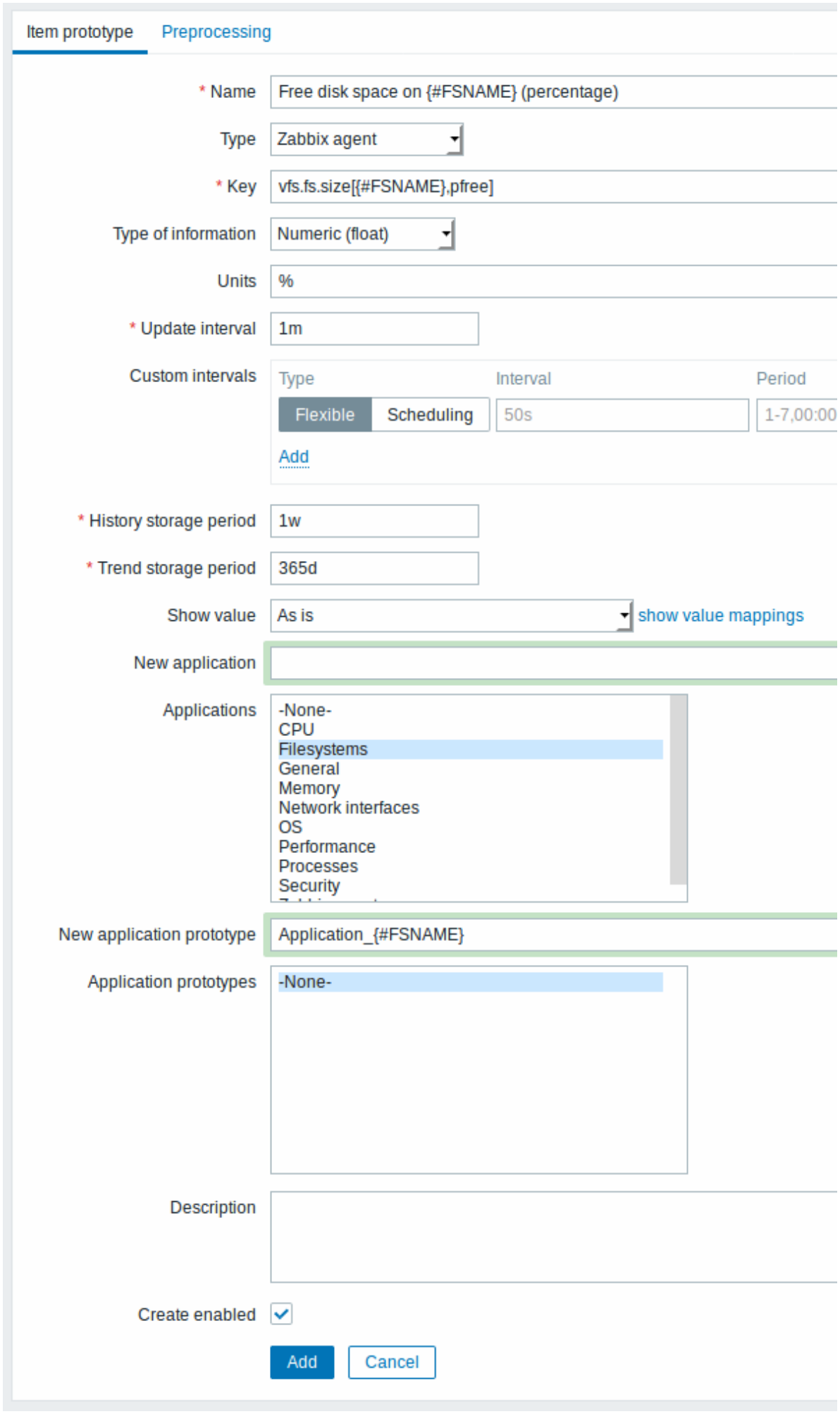
Кнопки диалога

Кнопки в нижней части диалога позволяют выполнить несколько видов операций.

Add	Добавление правила обнаружения. Эта кнопка доступна только для новых правил обнаружения.
Update	Обновление свойств правила обнаружения. Эта кнопка доступна только для уже существующих правил обнаружения.
Clone	Создание другого правила обнаружения на основе свойств текущего правила обнанужения.
Check now	Выполнение немедленного обнаружения на основе правила обнаружения. Правило обнаружения должно существовать. Смотрите более подробную информацию. *Обратите внимание*, что когда обнаружение выполняется немедленно, кэш конфигурации не обновляется, поэтому на результат не повлияют совсем недавние изменения настроек правила обнаружения.
Delete	Удаление правила обнаружения.
Cancel	тмена изменения свойств правила обнаружения.

Прототипы элементов данных

Как только правило будет создано, перейдем к элементам данных этого правила и нажмем “Создать прототип”, чтобы создать прототип элементов данных. Обратите внимание на то, как используется макрос {#FSNAME}, где требуется указать имя файловой системы. Когда правило будет обрабатываться, этот макрос будет заменен обнаруженной файловой системой.


Item prototype
Preprocessing
* Name
Free disk space on {#FSNAME} (percentage)
Type
Zabbix agent
* Key
vfs.fs.size[{#FSNAME},pfree]
Type of information
Numeric (float)
Units
%
* Update interval
1m
Custom intervals
Type
Interval
Period
Flexible
Scheduling
50s
1-7,00:00
Add
* History storage period
1w
* Trend storage period
365d
Show value
As is
show value mappings
New application
Applications
-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security
New application prototype
Application_{#FSNAME}
Application prototypes
-None-
Description
Create enabled
Add
Cancel

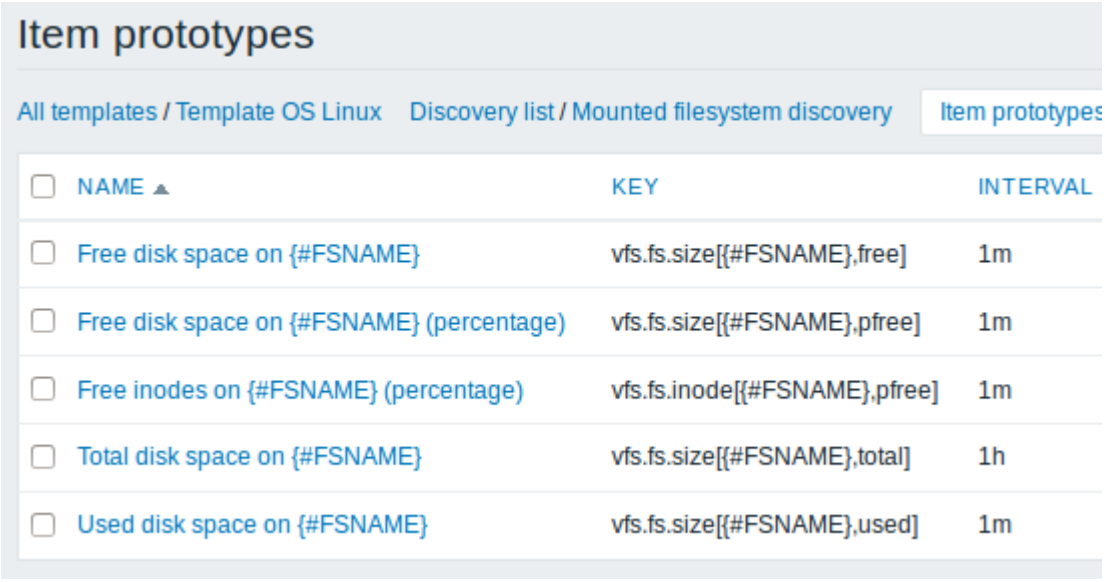
Можно использовать макросы низкоуровневого обнаружения и пользовательские макросы в настройках прототипа элементов данных и в параметрах предварительной обработки значений элемента данных.

Контекстное экранирование макросов низкоуровневого обнаружения для безопасного их использования в регулярных выражениях и параметрах предварительной обработки XPath.

Специфичные для прототипов элементов данных атрибуты:

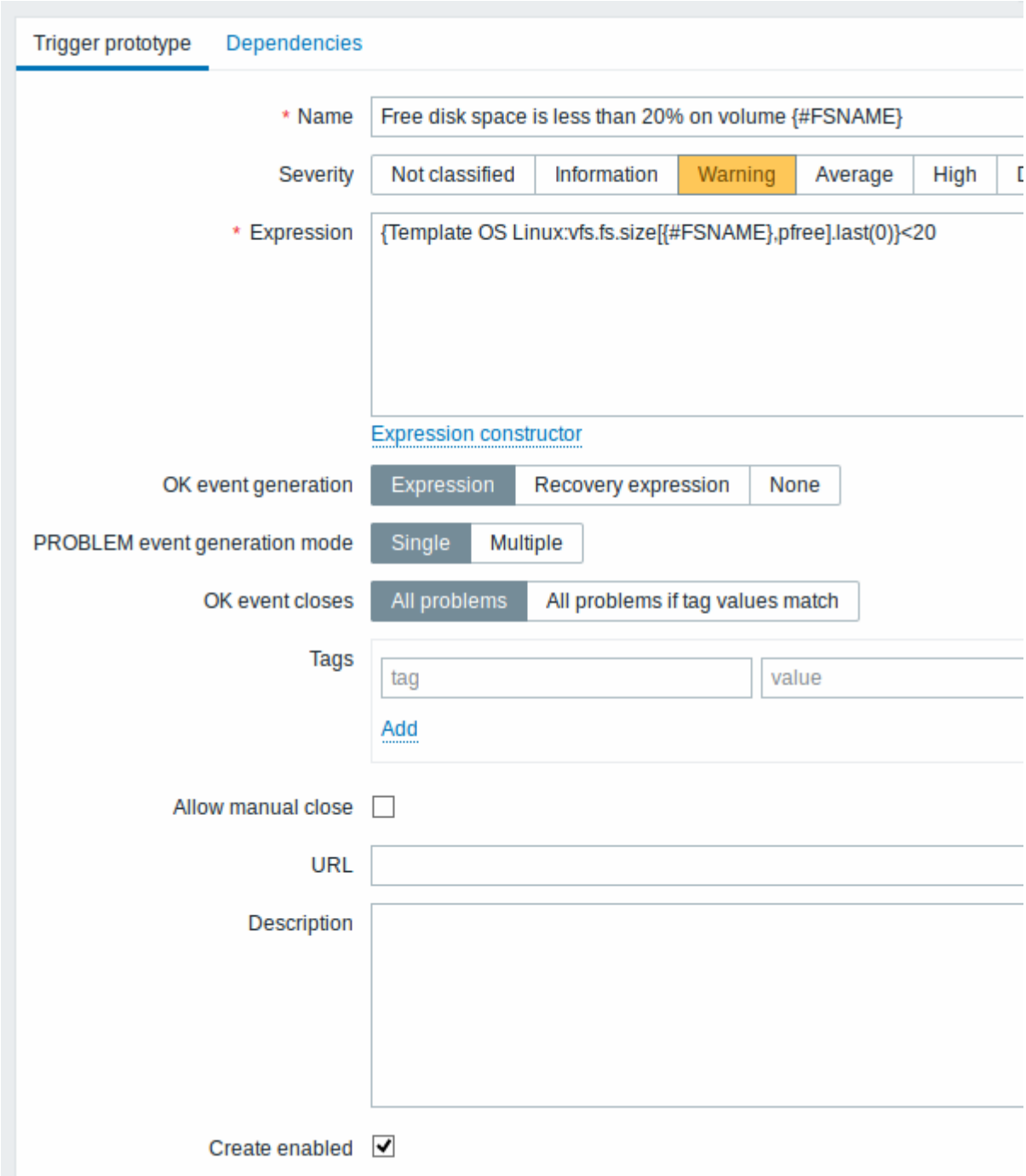
Параметр	**Описание**
Новый прототип группы элементов данных	Вы можете задать новый прототип группы элементов данных. В свойствах группы элементов данных вы можете использовать макросы низкоуровневого обнаружения, которые, после выполнения обнаружения, будут заменены реальными значениями при создании групп элементов данных, которые специфичны для обнаруженного объекта. Смотрите также заметки по обнаружению групп элементов данных для получения более подробной информации.
Прототипы групп элементов данных	Выберите из существующих прототипов групп элементов данных.
Создать активированным	Если выбрано, элемент данных будет создан в активированном состоянии. Если не выбрано, элемент данных будет добавлен как обнаруженный объект, но в деактивированном состоянии.

Мы можем создать несколько прототипов элементов данных для каждой интересующей нас характеристики файловой системы:



Прототипы триггеров

Мы создадим прототипы триггеров похожим способом как и прототипы элементов данных:



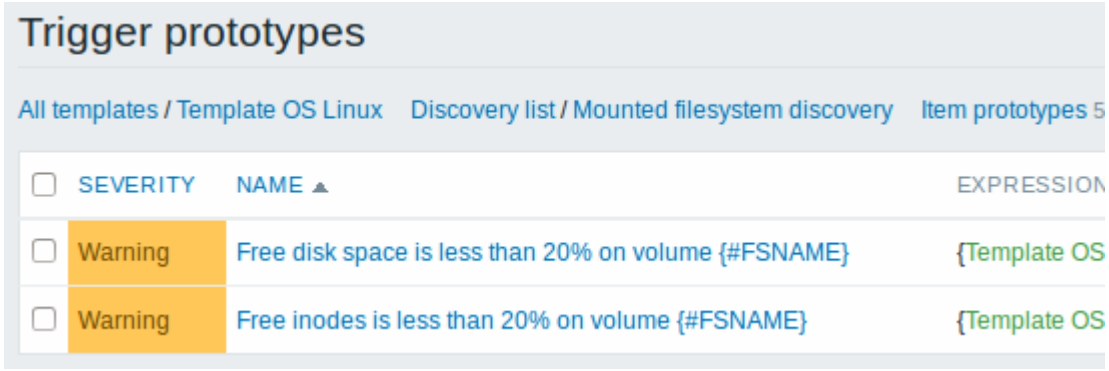
Специфичные для прототипов триггеров атрибуты:

Параметр	Описание
Создать активированным	Если выбрано, триггер будет создан в активированном состоянии. Если не выбрано, триггер будет добавлен как обнаруженный объект, но в деактивированном состоянии.

Когда будут созданы реальные триггера из их прототипов, возможно потребуется большая гибкость чем использованная константа ('20' в нашем примере) для сравнения в выражении. Смотрите каким образом пользовательские макросы с контекстом могут быть полезны для получения подобной гибкости.

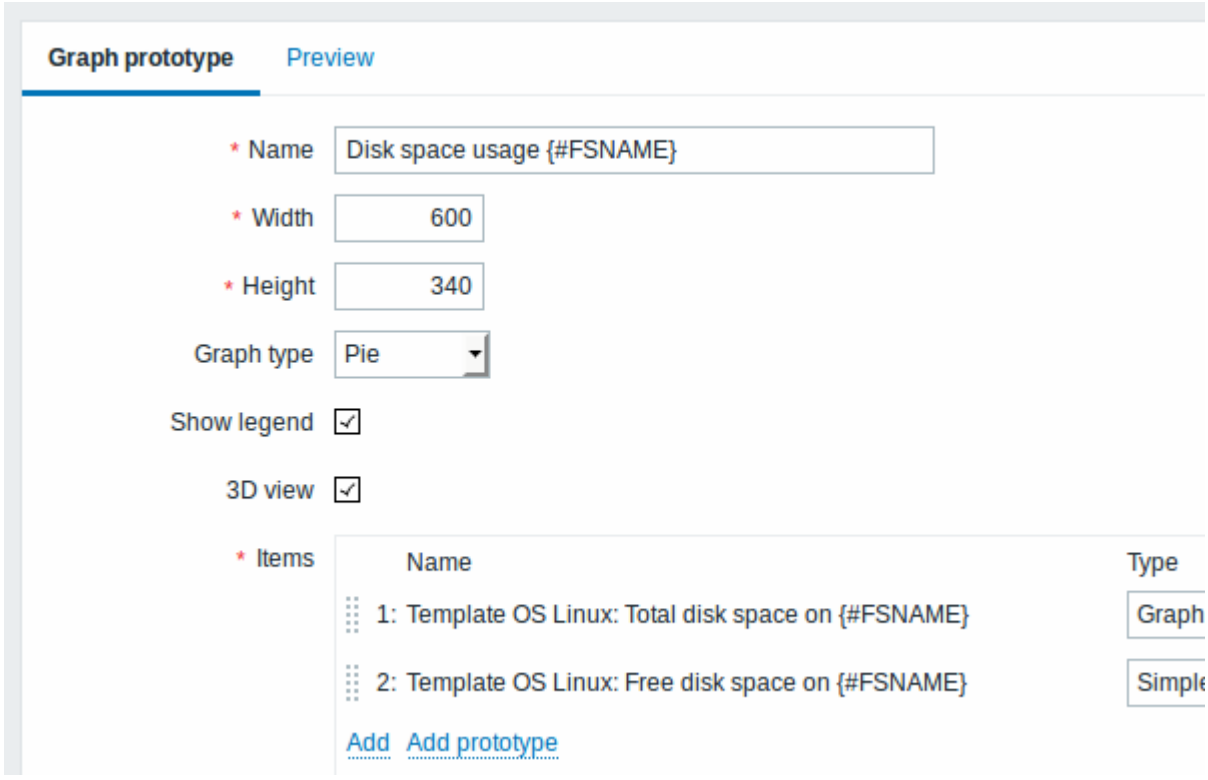
Также вы можете задать зависимости между прототипами триггеров (поддерживается начиная с Zabbix 3.0). Чтобы это сделать, перейдите на вкладку *Зависимости*. Прототип триггеров может зависеть от другого прототипа триггеров из этого же правила

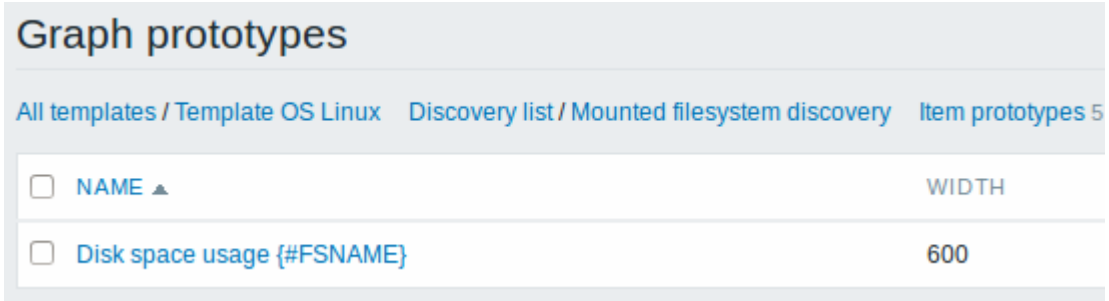
низкоуровневого обнаружения (LLD) или от обычного триггера. Прототип триггеров не может зависеть от прототипа триггеров из другого правила LLD и от триггера созданного другим прототипом триггеров. Прототип триггеров узла сети не может зависеть от триггера из шаблона.



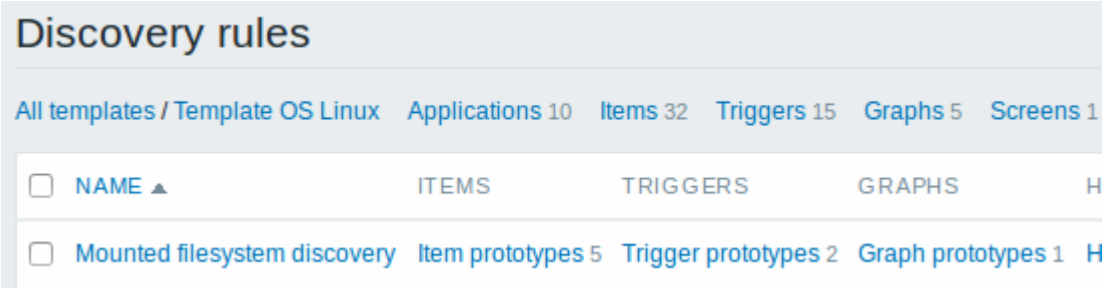
Прототипы графиков

Мы также можем создать прототипы графиков:





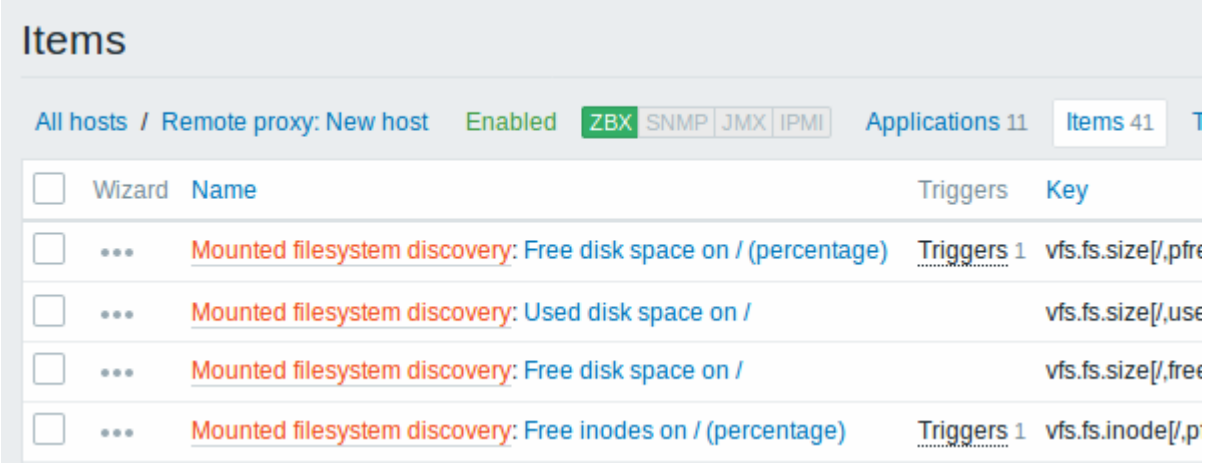
В конце концов, мы создали правило обнаружения, которое выглядит как видно ниже. Оно имеет пять прототипов элементов данных, два прототипа триггеров и один прототип графиков.



Обратите внимание: Для получения информации по настройке прототипов узлов сети, смотрите в разделе мониторинга виртуальных машин о настройке прототипов узлов сети.

Обнаруженные объекты

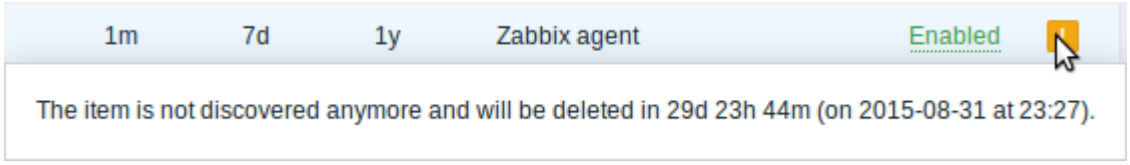
Представленные снимки экрана ниже иллюстрируют как выглядят уже обнаруженные элементы данных, триггера и графики в настройке узла сети. Обнаруженные объекты имеют префикс ссылку золотистого цвета, которая ведет к правилу обнаружения, создавшего эти объекты.



Обратите внимание, что обнаруженные объекты не будут созданы в случае, если объекты с такими же условиями уникальности уже существуют, например, элемент данных с таким же ключем или график с таким же именем.

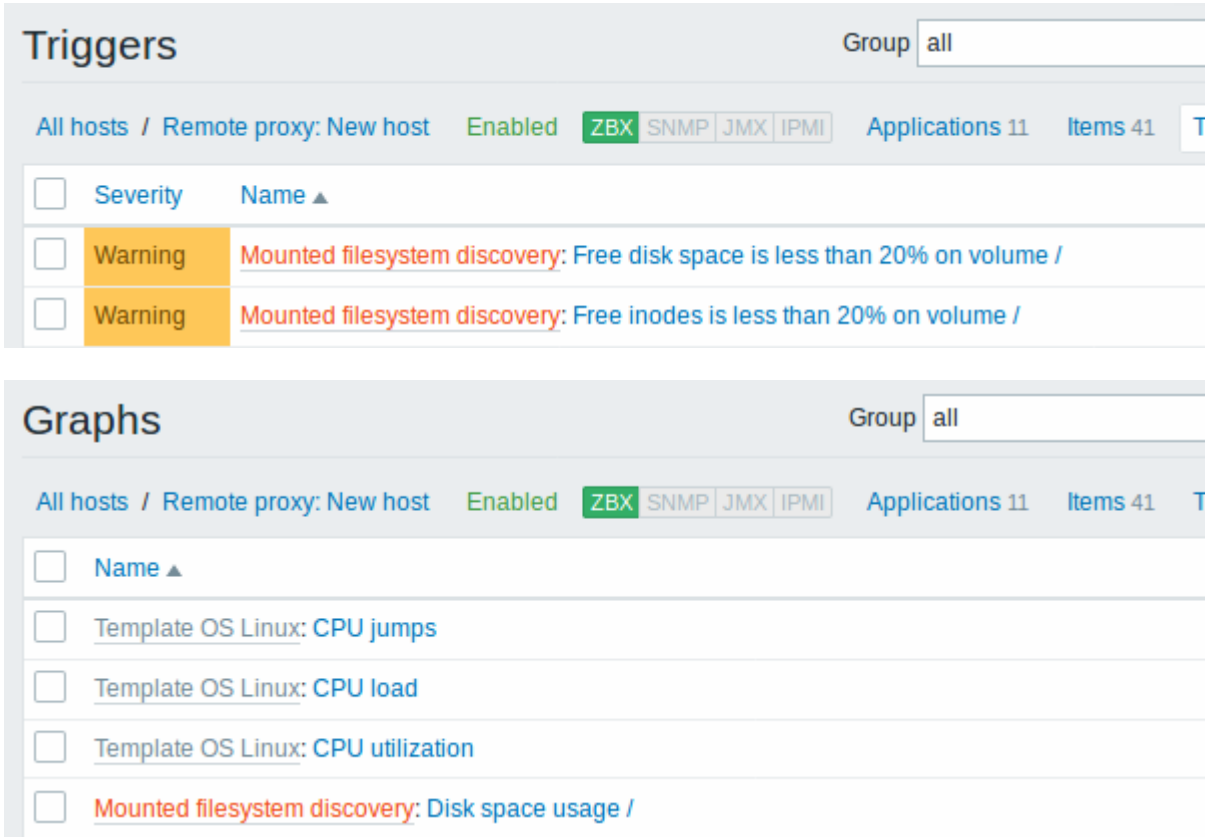
Элементы данных (а также, триггеры и графики) созданые с помощью низкоуровневого правила обнаружения невозможно удалить вручную. Тем не менее, они будут удалены автоматически, если обнаруженный объект (файловая система, интерфейс и т.д.) более не обнаруживается (или более не попадает под фильтр). В этом случае они будут удалены спустя некоторое количество дней указанное в поле *Период сохранения потерянных ресурсов*.

Когда обнаруженный объект становится 'Более не обнаруживается', в списке элементов данных будет отображаться оранжевый индикатор времени жизни. Переместите курсор мыши на этот индикатор и вы увидите сообщение с количеством дней до момента удаления элемента данных.



Если объекты помечены на удаление, но не были удалены в назначенное время (деактивировано правило обнаружения или элемент данных узла сети), они удалятся при следующем выполнении правила обнаружения.

Объекты, которые содержат другие объекты, которые помечены на удаление, не будут обновлены, если будут изменены на уровне правила обнаружения. Например, триггеры на основе LLD не будут обновлены, если они содержат элементы данных, которые помечены на удаление.



Другие типы обнаружения

Для получения более детальных сведений и инструкций по остальным типам доступных обнаружений сморите следующие разделы:

- обнаружение сетевых интерфейсов;
- обнаружение CPU и ядер CPU;
- обнаружение SNMP OID'ов;
- обнаружение JMX объектов;
- обнаружение с использованием ODBC SQL запросов;
- обнаружение Windows служб;
- обнаружение интерфейсов хостов в Zabbix.

Для получения более подробных сведений касательно JSON формата по обнаружению элементов данных и примера каким образом реализовать своё собственное обнаружение файловых систем при помощи Perl скрипта, смотрите создание пользовательских LLD правил.

Ограничения данных для возвращаемых значений

Ограничения для JSON данных низкоуровневого правила обнаружения отсутствуют, если эти данные получены напрямую Zabbix сервером, так как полученные значения обрабатываются без сохранения в базу данных. Также ограничения отсутствуют и для пользовательских правил низкоуровневого обнаружения, однако, если предполагается получение пользовательских LLD данных при помощи пользовательского параметра, тогда накладывается ограничение по размеру значения (512 КБ) на сам пользовательский параметр.

Если данные поступают от Zabbix прокси, этот прокси вынужден сначала записать их в базу данных. В таком случае накладываются ограничения к базе данных, например, 2048 байт для Zabbix прокси, который работает с IBM DB2 базой данных.

Несколько LLD правил по одному и тому же элементу данных

Начиная с Zabbix агента версии 3.2, имеется возможность задать несколько правил низкоуровневого обнаружения по одному и тому же элементу данных обнаружения.

Чтобы это сделать, вам необходио указать параметр агента Alias, разрешив использование измененных ключей элемента данных обнаружения в разных правилах обнаружения, например `vfs.fs.discovery[foo]`, `vfs.fs.discovery[bar]` и так далее.

1 Создание пользовательских LLD правил

Также имеется возможность создать полностью пользовательское правило низкоуровневого обнаружения, для обнаружения любого типа объектов - к примеру, баз данных на сервере баз данных.

Чтобы это сделать, необходимо создать пользовательский элемент данных, который будет возвращать JSON, определяющий найденные объекты и опционально - некоторые свойства этих объектов. Количество макросов на объект не ограничено - в то время как встроенные правила обнаружения возвращают либо один, либо два макроса (нппример, два в случае обнаружения файловых систем), имеется возможность возвращать больше.

Требуемый JSON формат лучше всего иллюстрируется в примере. Предположим, что мы оставим старый Zabbix агент версии 1.8 (который не поддерживает “vfs.fs.discovery”), но нам также нужно обнаруживать файловые системы. Вот простой Perl скрипт для Linux, который обнаруживает примонтированные файловые системы и выдает на выходе данные JSON, в которых включено и имя, и тип файловой системы. Одним из способов его использования является UserParameter с ключем “vfs.fs.discovery_perl”:

```
#!/usr/bin/perl
```

```
$first = 1;

print "{\n";
print "\t\"data\":[\n\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "\n\t]\n";
print "}\n";
```

Допустимыми символами в именах макросов низкоуровневых правил обнаружения являются **0-9** , **A-Z** , **_** , **.**

Буквы в нижнем регистре в именах не поддерживаются.

Пример его вывода (переформатирован для наглядности) представлен ниже. JSON данные от пользовательской проверки обнаружения следуют такому же формату.

```
{
  "data":[
  { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
  { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
  { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
  { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
  { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
  { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }
  ]
}
```

Тогда, в правилах обнаружения в поле “Фильтр” мы можем указать “{#FSTYPE}”, как макрос, и “rootfs|ext3”, как регулярное выражение.

Вы не обязаны использовать имена макросов FSNAME/FSTYPE в пользовательских правилах низкоуровневого обнаружения, вы можете использовать любые другие имена, которые вам нравятся.

Обратите внимание на то, что при использовании пользовательского параметра, возвращаемые данные ограничены 512 КБ. Для получения более подробных сведений смотрите ограничения данных для возвращаемых значений LLD.

2 Использование макросов LLD в контекстах пользовательских макросов

Пользовательские макросы с контекстом можно использовать для получения более гибких порогов в выражениях триггеров. Разные пороги можно задать на уровне пользовательского макроса и затем их можно использовать в константах триггеров, в зависимости от обнаруженного контекста. Обнаруженный контекст появляется, когда используемые макросы низкоуровневого обнаружения в макросах раскрываются в реальные значения.

Для иллюстрации мы можем использовать данные из приведенного примера выше, предположим, что будут обнаружены следующие файловые системы: `/`, `/home`, `/tmp`, `/usr`, `/var`.

Мы можем задать узлу сети прототип триггера на свободное место на диске, где порог выражается при помощи пользовательского макроса с контекстом:

`{host:vfs.fs.size[{#FSNAME},pfree].last()}<`**`{$LOW_SPACE_LIMIT:"{#FSNAME}"}`**

Затем добавим пользовательские макросы:

- `{$LOW_SPACE_LIMIT}` **10**
- `{$LOW_SPACE_LIMIT:/home}` **20**
- `{$LOW_SPACE_LIMIT:/tmp}` **50**

Тогда события сгенерируются, когда на файловых системах `/`, `/usr` и `/var` станет свободного места на диске меньше чем **10**%, файловой системе `/tmp` станет свободного места на диске менее чем **50**% или на файловой системе `/home` станет свободного места на диске менее чем **20**%.