Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

LLD-override Zabbix 7.4: почему старые Link down живы

Как мы гасим уже обнаруженные аварии портов, не трогая метрики интерфейсов

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

В LLD-override инженер ставит «Create enabled: No» для триггер-прототипа Link down и ждёт тишины, но старые алерты продолжают сыпаться. Операции override применяются в момент создания сущности, поэтому ранее обнаруженные триггеры остаются в статусе Enabled. Инженер либо удаляет метрики интерфейсов руками, теряя историю трафика, либо мирится с шумом на проде.

Почему это важно бизнесу

- Ложные ночные звонки по портам, которые заказчик сам считает неважными, обесценивают весь мониторинг в глазах клиента
- Удаление item'ов интерфейса вручную стирает историю net.if.in/net.if.out и счётчиков — данные для расследования инцидента пропадают безвозвратно
- Каждая правка override без понимания lifetime отнимает у инженера 20-40 минут на диагностику «почему не подействовало»
- Шумные алерты приучают дежурного игнорировать Link down, и он пропускает реальный обрыв аплинка

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

7.4

версия Zabbix, где LLD overrides и раздельные Delete/Disable lost resources штатны

Zabbix 7.4, Low-level discovery

No

значение Create enabled в override — задаёт статус создаваемых сущностей, уже созданные не тро…

Zabbix 7.4, LLD overrides

1 ч

наш enabled_lifetime (Disable lost resources: After) вместо дефолтного Immediately

стандарт ITfresh

30 д

наш lifetime (Delete lost resources: After) вместо дефолтных 7d — история графиков переживает…

стандарт ITfresh

0/1

значения макроса {$IFCONTROL:"{#IFNAME}"} — точечно гасим Link down без правки шаблона

Zabbix 7.4, Interfaces by SNMP

^2$

{$NET.IF.IFADMINSTATUS.NOT_MATCHES} — LLD-фильтр отсекает порты в ifAdminStatus down(2)

Zabbix 7.4, шаблоны сетевых устройств

РЕАЛИЗАЦИЯ 1 ИЗ 3

Почему override не гасит уже сработавшие Link down

Что настраиваем

Правило для всех LLD-правил обнаружения интерфейсов на сетевых узлах клиента

Как мы это делаем

1. В Data collection → Hosts → Discovery открываем override, в Operations находим условие по триггер-прототипу Link down и видим Create enabled: No
2. Сверяем с документацией: операции override задают свойства создаваемых объектов, то есть флаг отрабатывает на обнаружении, а статус ранее созданных триггеров не пер…
3. Уже существующие Link down по ранее найденным {#SNMPINDEX} остаются Enabled и продолжают переходить в Problem — это не баг, а ожидаемая семантика
4. Гасим их отдельно: Data collection → Hosts → Triggers, фильтр по шаблону/тегу, выделяем список и жмём Disable под таблицей, либо тем же отбором через API trigger.up…
5. Проверяем результат через discoveryrule.get и triggerprototype.get: новых Link down не создаётся, а старые сняты с боевого дежурства без удаления item'ов

РЕЗУЛЬТАТ

Шум по неважным портам исчезает в течение одного цикла discovery, а история net.if.in/net.if.out и net.if.status остаётся нетронутой — заказчик не теряет данные для последующего разбора инцидентов.

КЛЮЧЕВОЙ НЮАНС

Create enabled — это статус для вновь создаваемых сущностей, а не команда «отключить всё, что уже есть»: для обнаруженных ранее триггеров нужен отдельный шаг Disable в списке или API trigger.update.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Управляем жизненным циклом портов вместо ручного удаления

Что настраиваем

Настройка discovery rule на шаблоне-модуле «Interfaces by SNMP»

Как мы это делаем

1. На уровне LLD-правила ставим Disable lost resources = After и enabled_lifetime = 1h — порт отключается через час, а не мгновенно, как в дефолте (enabled_lifetime_ty…
2. Отдельно задаём Delete lost resources = After, lifetime = 30d вместо дефолтных 7d — item и график удаляются, только когда точно ясно, что порт демонтирован физически
3. Значение Delete lost resources: Immediately не используем: по документации ошибочная правка фильтра тогда стирает сущность вместе со всей историей
4. Для критичных аплинков переопределяем параметры через override-фильтр по {#IFNAME} или {#IFALIAS}, не трогая остальные 40+ портов коммутатора
5. Раз в квартал сверяем список disabled-но-не-deleted интерфейсов с реальной схемой коммутации — исключаем накопление мусора без потери истории

РЕЗУЛЬТАТ

Пропавший порт не улетает в удаление молча: час на подтверждение, тридцать дней на историю графиков — этого хватает и инженеру, и клиенту, чтобы среагировать до потери данных.

КЛЮЧЕВОЙ НЮАНС

Delete lost resources и Disable lost resources — разные таймеры, причём период удаления обязан быть больше периода отключения; путать их — то удалить нужную историю раньше времени, то держать мёртвые триггеры…

РЕАЛИЗАЦИЯ 3 ИЗ 3

Точечная тишина по шумным портам через {$IFCONTROL}

Что настраиваем

Массовое подавление Link down на access-портах без правки триггер-прототипа

Как мы это делаем

1. Штатный триггер-прототип содержит условие {$IFCONTROL:"{#IFNAME}"}=1 and last(/…/net.if.status[ifOperStatus.{#SNMPINDEX}])=2 and (last(…,#1)<>last(…,#2)) — это офиц…
2. На хосте задаём макрос {$IFCONTROL:"Gi1/0/24"} = 0 для access-порта с принтером — Link down не создаёт Problem, item со счётчиками пишет историю дальше
3. Массово раскладываем макросы через host prototype в LLD хостов или через API hostmacro.create/hostmacro.update — без правки самого шаблона и без override под каждый…
4. Дополнительно закрываем административно выключенные порты: в фильтре LLD-правила {#IFADMINSTATUS} с оператором does not match и макросом {$NET.IF.IFADMINSTATUS.NOT_…

РЕЗУЛЬТАТ

Инженер отключает шум по конкретному порту за один макрос вместо правки override или прототипа — изменение точечное, обратимое и видно в списке макросов хоста.

КЛЮЧЕВОЙ НЮАНС

Не каждое подавление алерта требует override: если в шаблоне уже есть управляющий макрос вроде {$IFCONTROL}, это быстрее и безопаснее правки прототипов.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Ждём, что override уберёт старые алерты**
  Create enabled задаёт статус создаваемых сущностей; уже обнаруженные триггеры гасим отдельно — кнопкой Disable в списке Triggers или API trigger.upda…

- ✕ **Удаляем item интерфейса вручную**
  Теряем историю net.if.in/net.if.out навсегда; вместо удаления ставим Disable lost resources: After с enabled_lifetime порядка 1h.

- ✕ **Delete lost resources: Immediately**
  Порт с историей исчезает сразу, как только LLD его не увидел; документация прямо не рекомендует этот режим — ошибка в фильтре стирает данные.

- ✕ **Правим шаблон вместо override/макроса**
  Изменение прототипа задевает все хосты, привязанные к шаблону; точечный кейс решаем override-фильтром по {#IFNAME} или макросом {$IFCONTROL}.

- ✕ **Самодельный триггер без сравнения последних…**
  Функция diff() удалена начиная с Zabbix 5.4; смену статуса ловим конструкцией last(item,#1)<>last(item,#2), иначе триггер реагирует на каждый опрос.

- ✕ **Игнорируем ifAdminStatus**
  Административно выключенный порт (ifAdminStatus = down(2)) не авария; без фильтра {$NET.IF.IFADMINSTATUS.NOT_MATCHES} = ^2$ дежурный получает ложный…

- ✕ **Отключаем триггеры по одному в GUI**
  На сотне портов это часы работы; фильтруем список Triggers по шаблону/тегу и жмём Disable либо шлём API trigger.update с массивом triggerid.

- ✕ **Не проверяем discoveryrule после правки**
  Изменённые lifetime, enabled_lifetime и overrides перепроверяем через discoveryrule.get — эффективные параметры удобнее читать в JSON, чем в форме GU…

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Отключаем уже созданные Link down: фильтр списка Triggers по шаблону и кнопка Disable
- Не удаляем item'ы интерфейсов вручную — только Disable, история net.if остаётся
- Считаем Create enabled в override правилом для вновь создаваемых сущностей, а не вык…

НОРМАЛЬНО

- Ставим Disable lost resources: After с enabled_lifetime 1h вместо дефолтного Immedia…
- Вводим {$IFCONTROL:"{#IFNAME}"}=0 на хостах с шумными access-портами вместо правки п…
- Фильтруем {#IFADMINSTATUS} по {$NET.IF.IFADMINSTATUS.NOT_MATCHES} = ^2$ на уровне LL…
- Раз в квартал сверяем список disabled-интерфейсов со схемой коммутации

ХОРОШО

- Автоматизируем массовое управление триггерами через API trigger.update с отбором по…
- Раздельно тюним lifetime (Delete, 30d) и enabled_lifetime (Disable, 1h) под критично…
- Документируем override-политику LLD в runbook команды с примерами условий и операций
- Регулярно снимаем discoveryrule.get и сравниваем конфигурацию LLD между хостами шабл…

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Проверено, что Create enabled: No задаёт статус только создаваемых сущностей?
- [ ] Старые Link down по ранее обнаруженным портам сняты кнопкой Disable или API trigger.update?
- [ ] Метрики интерфейсов не удалялись вручную, история net.if сохранена?
- [ ] На LLD-правиле разделены Disable lost resources (1h) и Delete lost resources (30d)?
- [ ] На шумных портах задан макрос {$IFCONTROL:"{#IFNAME}"}=0 вместо правки шаблона?
- [ ] В фильтре LLD учтён {#IFADMINSTATUS} с {$NET.IF.IFADMINSTATUS.NOT_MATCHES} = ^2$?
- [ ] В самодельных триггерах смена статуса ловится через last(item,#1)<>last(item,#2), а не удалённой diff()?
- [ ] Изменения override и lifetime перепроверены через discoveryrule.get?
- [ ] Есть runbook с шагами на случай «override не подействовал на старые сущности»?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Настраиваем LLD-правила и override для сетевых шаблонов под конкретную схему коммутации заказчика
- Разводим Disable/Delete lost resources и {$IFCONTROL} так, чтобы шум уходил, а история не терялась
- Массово приводим в порядок уже созданные триггеры через API trigger.update без ручной правки по одному
- Пишем runbook и передаём внутренней ИТ-команде клиента логику override для самостоятельной поддержки
- Аудитируем существующие шаблоны Zabbix на предмет забытых Create enabled и режима Delete lost resources: Immediate…

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Low-level discovery — обзор, lifetime и lost resources (zabbix.com — 7.4)

02 Low-level discovery — Overrides (условия и операции) (zabbix.com — 7.4)

03 Notes on low-level discovery (zabbix.com — 7.4)

04 LLD rule object — lifetime_type, enabled_lifetime_type (API reference) (zabbix.com — 7.4)

05 Trigger prototypes (zabbix.com — 7.4)

06 Standardized templates for network devices (Interfaces by SNMP) (zabbix.com — 7.4)

07 trigger.update (API reference) (zabbix.com — 7.4)

08 History functions (change, last) — приложение по функциям (zabbix.com — 7.4)

09 Внутренний стандарт ITfresh по LLD сетевых устройств (runbook) (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.