ТЕХНИЧЕСКИЙ ОТЧЁТ

Пропуски срабатывания автономных DALI-клавиш при активном сервисе wb-mqtt-dali

Причина, локальный workaround, проверка результата и предложение для upstream-исправления

Установленная версия: wb-mqtt-dali 1.19.4
Оборудование: Wiren Board + WB-DALI-3, 3 линии DALI
Внешние управляющие устройства: Skydance DA-4S
Дата фиксации результата: 06.08.2026

Подготовлено для технической поддержки и разработчиков Wiren Board

Статус: подтверждено сравнительным тестом до/после правки. Отдельно отмечены факты, выводы и ограничения.

Содержание

1. Резюме
2. Исходная конфигурация
3. Симптомы и сравнительный тест
4. Что происходило в коде и в шине
5. Подтверждённая причина
6. Внесённые изменения
7. Проверка результата
8. Ограничения workaround
9. Рекомендации для официального исправления
10. Точный журнал действий
11. Источники и уровень уверенности

Ключевая корректировка предыдущей формулировки

Физическая потеря DALI-кадра не была подтверждена осциллографом или независимым сниффером. Подтверждено другое: активный wb-mqtt-dali создавал дополнительный трафик после внешних команд; после отключения механизма confirmation polls пропуски исчезли. Поэтому ниже используется аккуратная формулировка «наиболее вероятный механизм - конкуренция/коллизии и занятость шины», а не недоказанное утверждение о конкретном месте потери кадра.

1. Резюме

Проблема. При запущенном сервисе wb-mqtt-dali автономные настенные DALI-модули Skydance DA-4S периодически давали пропуски, задержки или требовали повторного нажатия. При остановленном сервисе те же клавиши работали стабильно.

Наблюдение в коде. Сервис обрабатывает как собственные, так и увиденные на шине внешние команды одним методом EventSyncCoordinator.apply_commands(). Для команд изменения уровня он оптимистично обновляет состояние MQTT, а затем вызывает _schedule_confirm(), который планирует последующий подтверждающий опрос затронутых параметров.

Локальное решение. В /usr/lib/python3/dist-packages/wb/mqtt_dali/event_sync_coordinator.py в начале метода _schedule_confirm() добавлен немедленный return. Это полностью отключило confirmation polls - как для внешних, так и для собственных команд.

Результат. После синтаксической проверки, перезапуска сервиса и завершения фонового startup-fetch все проверенные DA-4S стали работать без пропусков и задержек. Сервис wb-mqtt-dali остался активным.

Вывод с высокой степенью уверенности

Практическая причина нестабильности - активный дополнительный трафик, запускаемый механизмом подтверждения после внешних DALI-команд. На данном крупном объекте этот трафик конфликтовал с автономным управлением. Полное отключение confirmation polls устранило проблему в повторном полевом тесте.

2. Исходная конфигурация

Параметр	Значение
Контроллер/шлюз	Wiren Board, модуль WB-DALI-3

Параметр	Значение
Количество шин	3 независимые линии DALI
Сервис	wb-mqtt-dali 1.19.4
Масштаб	Более 100 control gear суммарно по трём линиям
Типы нагрузок	Диммируемые драйверы и большое количество релейных устройств, включая DT7
Настенные модули	Skydance DA-4S, работающие автономно на DALI-шине
Роль DA-4S	Напрямую формируют forward-команды для control gear; в MQTT как DALI-2 Input Device не публикуются
Пример адреса	A28: кадры 0x3800 и 0x38FE соответствуют DAPC уровням 0 и 254

Существенная особенность архитектуры: DA-4S не должны зависеть от Wiren Board для основного ручного управления. Их штатный сценарий - отправить DALI-команду непосредственно исполнительному устройству. wb-mqtt-dali в этой цепочке нужен для интеграции и синхронизации состояния, но не для факта срабатывания клавиши.

3. Симптомы и сравнительный тест

3.1. Симптомы при активном сервисе

- отдельные нажатия DA-4S не приводили к ожидаемому переключению;
- иногда требовалось повторное нажатие;
- наблюдались задержки включения/выключения;
- в журнале фиксировались Broken response, missing и framing error;
- после внешних level-команд появлялись дополнительные запросы состояния со стороны wb-mqtt-dali.

3.2. A/B-проверка

Параметр	Значение
Сервис остановлен	DA-4S работают стабильно; пропуски исчезают.
Сервис запущен без финальной правки	Пропуски воспроизводятся.
Сервис запущен после отключения _schedule_confirm()	После завершения startup-fetch все проверенные зоны работают идеально.

Что доказывает тест. Источник нестабильности находится в активности wb-mqtt-dali либо в её взаимодействии с большим объёмом внешнего трафика. Это не исключает чувствительность конкретных устройств к таймингам DALI, однако исключает объяснение только неисправностью DA-4S, кабеля или питания: при той же физической линии и остановленном сервисе проблема исчезала.

4. Что происходило в коде и в шине

4.1. Обработка внешней команды

Официальный исходный код описывает EventSyncCoordinator как слой, который превращает команды, замеченные на шине («sniffed + our own»), в оптимистические обновления MQTT и подтверждающие опросы. Метод apply_commands() получает только список Command и не различает происхождение команды на уровне своего интерфейса.

Упрощённый фрагмент логики

```
async def apply_commands(self, commands: list[Command]) -> None:
    """Apply a burst of bus commands (sniffed or our own)
    to MQTT state + confirm polls."""
    for command in commands:
        await self._apply_one(command)
```

4.2. Команда уровня и планирование подтверждения

Для DAPC/Off/RecallMaxLevel и других level-команд _apply_level() обновляет предсказанное состояние ActualLevel, добавляет ActualLevelControl в список pollables, а для DT7 также может добавить LastActedControl. После публикации оптимистического состояния вызывается _schedule_confirm().

Смысловой фрагмент _apply_level()

```
pollables = [actual]
last_acted = device.get_mqtt_control("last_acted")
if isinstance(last_acted, LastActedControl):
    pollables.append(last_acted)
...
self._schedule_confirm(device, pollables, basis)
```

4.3. Что делает _schedule_confirm()

Метод вычисляет момент, когда устройство должно выйти на установившееся состояние, и вызывает schedule_confirmation() для каждого затронутого pollable. В дальнейшем планировщик выполняет реальный DALI-запрос соответствующего состояния.

Официальная логика планирования

```
def _schedule_confirm(self, device, pollables, basis) -> None:
    settle = self._settle.settle_for(basis, device.fade_param.fade_time)
    now = self._now()
    at = now + settle
    for pollable in pollables:
        pollable.schedule_confirmation(now, at)
```

4.4. Фактическая последовательность в журнале

До правки	После правки
Внешний кадр DA-4S 0x3800 / 0x38FE Затем: QueryActualLevel(A28) DT7.QuerySwitchStatus(A28) (и служебное включение DT7, если требуется)	Внешний кадр DA-4S 0x3800 / 0x38FE После правки: нет связанного QueryActualLevel нет связанного DT7.QuerySwitchStatus

Наиболее вероятный механизм пропуска. На крупной линии подтверждающие запросы, ожидание backward-ответов и повторы при ошибках увеличивали занятость шины. Следующая внешняя команда могла приходиться на неблагоприятный момент. Это объяснение согласуется с журналом и A/B-тестом, но точное физическое место конфликта не фиксировалось независимым анализатором шины.

4.5. Отдельный startup-fetch

После старта сервиса дополнительно запускается SettingsFetchScheduler - фоновое инкрементальное чтение статических параметров. В его список входят MaxLevel, MinLevel, сцены и пороги DT7. Именно этим объясняются QueryMaxLevel, QuerySceneLevel и QueryUp/DownSwitch...Threshold, наблюдавшиеся сразу после перезапуска.

Состав фонового startup-fetch

```
FETCH_PARAM_CLASSES = (
    MaxLevelParam, MinLevelParam, ScenesParam, ScenesSettings,
    UpSwitchOnThresholdParam, UpSwitchOffThresholdParam,
    DownSwitchOnThresholdParam, DownSwitchOffThresholdParam,
)
```

> **Почему пропуски были видны, пока шёл journalctl -f**
>
> journalctl не обращается к DALI-шине. Он лишь показывал журнал. Совпадение объясняется тем, что после перезапуска ещё продолжался startup-fetch с запросами и повторами. К моменту закрытия просмотра этот обход в основном завершился.

5. Подтверждённая причина

Корневая архитектурная проблема: в EventSyncCoordinator внешние команды, пассивно замеченные в bus traffic, и команды самого wb-mqtt-dali проходят через общий путь, который безусловно планирует confirmation polls.

> **Что подтверждено напрямую**
>
> 1) apply_commands() официально предназначен для «sniffed or our own» команд; 2) _apply_level() всегда вызывает _schedule_confirm() для затронутых control gear; 3) после локального return в _schedule_confirm() связанные QueryActualLevel/DT7.QuerySwitchStatus исчезли; 4) после этого полевой тест прошёл без пропусков.

Что остаётся выводом, а не прямым измерением: конкретный электрический/временной механизм каждого пропущенного нажатия. Наиболее правдоподобно, что дополнительный активный трафик и повторы создавали коллизии или лишнюю занятость шины. Для окончательной протокольной локализации понадобился бы независимый DALI-анализатор с временными метками.

6. Внесённые изменения

6.1. Основная правка, устранившая пропуски

Файл: /usr/lib/python3/dist-packages/wb/mqtt_dali/event_sync_coordinator.py

Метод: EventSyncCoordinator._schedule_confirm()

Изменение: немедленный выход из метода до вычисления settle и постановки pollables в очередь подтверждения.

Фактически применённый diff

```
@@
 def _schedule_confirm(...):
+    return  # LOCAL_DISABLE_DALI_CONFIRM_POLLS
     settle = self._settle.settle_for(basis, device.fade_param.fade_time)
```

Команда, которой внесена правка:

```
python3 -c 'from pathlib import Path; p=Path("/usr/lib/python3/dist-packages/wb/mqtt_dali/event_sync_coordinator.py");
s=p.read_text(); old="    ) -> None:\n        settle = self._settle.settle_for(basis, device.fade_param.fade_time)\n"; new="
) -> None:\n        return  # LOCAL_DISABLE_DALI_CONFIRM_POLLS\n        settle = self._settle.settle_for(basis,
device.fade_param.fade_time)\n"; assert old in s, "МЕСТО ДЛЯ ПАТЧА НЕ НАЙДЕНО"; p.write_text(s.replace(old,new,1));
print("PATCHED")'
```

Результат команды: PATCHED

6.2. Проверка синтаксиса и перезапуск

```
python3 -m py_compile /usr/lib/python3/dist-packages/wb/mqtt_dali/event_sync_coordinator.py
# вывод отсутствовал - синтаксис корректен

systemctl restart wb-mqtt-dali
systemctl is-active wb-mqtt-dali
# active
```

6.3. Резервная копия

Ранее была создана резервная копия: /root/event_sync_coordinator.py.backup

6.4. Предварительная настройка интервала, не решившая проблему

На контроллере уже было установлено:

```
EVENT_RESYNC_BASE_INTERVAL = 86400.0  # LOCAL_DALI_LOW_TRAFFIC
# файл: /usr/lib/python3/dist-packages/wb/mqtt_dali/common_dali_device.py
```

Это уменьшало редкую фоновую ресинхронизацию, но не отключало event-driven confirmation polls, которые создавались непосредственно после каждой замеченной level-команды. Поэтому само по себе изменение интервала проблему не решило.

6.5. Сопутствующая локальная обработка monitor frame

В журнале также видно действие отдельной ранее внесённой защиты в wbdali.py: 16-битный кадр 0x3800/0x38FE, ошибочно помеченный шлюзом как backward, распознаётся как forward. Сообщение имеет вид:

```
Invalid backward monitor frame: length=16 data=0x3800; treating as forward
```

Эта правка важна для корректного декодирования внешних кадров и предотвращения ошибки «Initial data will not fit in 8 bits», но по результату тестов именно она не была финальным решением пропусков. Финальный эффект получен после отключения _schedule_confirm().

7. Проверка результата

7.1. Проверка журнала

После перезапуска внешние кадры DA-4S продолжали распознаваться. При этом связанная последовательность QueryActualLevel/DT7.QuerySwitchStatus после них больше не появлялась.

7.2. Полевой тест

- проверены разные зоны и клавиши;
- проверены последовательные включения/выключения;
- после завершения startup-fetch пропуски не воспроизводятся;
- задержки и необходимость двойного нажатия исчезли;
- wb-mqtt-dali продолжает работать и имеет состояние active.

> **Итог теста пользователя**
>
> «Все работает идеально».

8. Ограничения workaround

Добавленный return - намеренно грубый локальный workaround. Он отключает confirmation polls глобально, не только для внешних DA-4S-команд.

- после собственной команды wb-mqtt-dali сервис также не выполняет немедленное подтверждающее чтение;
- если control gear не исполнил команду, оптимистическое MQTT-состояние может временно не совпасть с фактическим;
- ошибка может быть выявлена только поздней ресинхронизацией или другим периодическим опросом;
- правка находится внутри установленного Python-пакета и может быть перезаписана обновлением wb-mqtt-dali;
- после обновления пакета необходимо проверить наличие/отсутствие LOCAL_DISABLE_DALI_CONFIRM_POLLS.

> **Эксплуатационная оценка**
>
> Для данного объекта надёжность автономных настенных клавиш важнее немедленного read-back после каждой команды. Поэтому компромисс приемлем как временное решение, но не должен становиться универсальным поведением по умолчанию.

9. Рекомендации для официального исправления

9.1. Разделить команды по источнику

Наиболее чистое решение - передавать в EventSyncCoordinator источник команды и применять разные политики подтверждения.

```
async def apply_commands(commands, source):
    ...
    if source is LOCAL_TX:
        schedule_confirmation()
    elif source is BUS_MONITOR:
        publish_optimistic_state_only()
```

В текущей архитектуре нижележащая очередь передачи уже содержит source, однако EventSyncCoordinator.apply_commands() принимает только список Command. Поэтому источник следует протянуть до слоя event sync либо разделить entry point для собственных и sniffed-команд.

9.2. Безопасные варианты поведения для внешней команды

- по умолчанию не выполнять немедленный confirmation poll для команды из bus monitor;
- либо подтверждать только после периода тишины на шине;
- отменять/переносить отложенное подтверждение при появлении нового внешнего кадра;
- объединять несколько быстрых команд к одному адресу в одно подтверждение последнего состояния;
- дать пользователю настройку confirm_external_commands=false или bus_monitor_mode=passive.

9.3. Приоритеты и backoff

- не выполнять низкоприоритетный confirmation poll во время внешнего трафика;
- после framing error/missing использовать увеличивающийся backoff, а не немедленную серию повторов;
- приостанавливать startup-fetch при обнаружении активности другого application controller;
- ограничить число запросов на одну idle-итерацию и предусмотреть минимальную паузу между пакетами.

9.4. Оптимизация startup-fetch

- кэшировать статические параметры между перезапусками;
- не перечитывать все 16 сцен каждого устройства без необходимости;
- ввести настраиваемую скорость фонового чтения;

- останавливать fetch на период внешнего пользовательского трафика;
- в диагностике явно разделять startup-fetch и confirmation poll.

9.5. Минимальный критерий приемки исправления

Параметр	Значение
Тестовая линия	Не менее 50 control gear, включая несколько DT7, плюс внешние автономные command devices.
Нагрузка	Серия быстрых внешних DAPC/Off/RecallMaxLevel при одновременно активном wb-mqtt-dali.
Критерий	Нет пропущенных пользовательских команд; confirmation/startup traffic не блокирует внешний трафик.
Контроль	Сравнение с остановленным сервисом и независимый DALI trace с временными метками.

10. Точный журнал действий

Шаг	Команда/действие	Результат
Проверили интервал событийной ресинхронизации	`grep -Rni "EVENT_RESYNC_BASE_INTERVAL" /usr/lib/python3/dist-packages/wb/mqtt_dali/common_dali_device.py`	EVENT_RESYNC_BASE_INTERVAL = 86400.0
Подтвердили наличие DT7-обработчиков	`grep -Rni "DaliType7|Type7|dali_type7" /usr/lib/python3/dist-packages/wb/mqtt_dali 2>/dev/null`	event_sync_coordinator.py импортирует LastActedControl; dali_device.py регистрирует Type7Parameters
Исследовали фоновый fetch	`sed -n "20,45p" /usr/lib/python3/dist-packages/wb/mqtt_dali/fetch_scheduler.py`	Выявлены Max/MinLevel, Scenes и DT7 thresholds
Нашли PollScheduler	`sed -n "273,350p" /usr/lib/python3/dist-packages/wb/mqtt_dali/application_controller.py`	Планировщик опрашивает poll_controls каждого initialized device
Внесли основной workaround	`Правка event_sync_coordinator.py: return в начале _schedule_confirm()`	PATCHED
Проверили синтаксис	`python3 -m py_compile .../event_sync_coordinator.py`	Без вывода, успешно
Перезапустили сервис	`systemctl restart wb-mqtt-dali`	Без вывода, успешно
Проверили состояние	`systemctl is-active wb-mqtt-dali`	active
Проверили лог и работу клавиш	`journalctl -u wb-mqtt-dali -f -n 0 + ручной тест`	После startup-fetch все клавиши работают идеально

11. Источники и уровень уверенности

11.1. Что перепроверено по официальному репозиторию

- README, раздел «Обновление состояния»: сервис обновляет состояние по чужим командам и после такой команды выполняет однократный подтверждающий опрос;
- event_sync_coordinator.py: слой обрабатывает «sniffed + our own» команды, _apply_level() формирует pollables и вызывает _schedule_confirm();
- event_sync_coordinator.py: _schedule_confirm() вызывает pollable.schedule_confirmation(now, at);
- fetch_scheduler.py: startup/background fetch включает MaxLevel, MinLevel, Scenes и пороги DT7;
- wbdali.py: source существует в элементах очереди передачи, что подтверждает техническую реализуемость разделения политик по источнику.

Официальный источник: публичный репозиторий wirenboard/wb-mqtt-dali, ветка main, проверено 06.08.2026. Установленная на объекте версия - 1.19.4; локальная структура методов совпала с проверенной логикой, что подтверждается успешным точечным патчем.

11.2. Материалы конкретного объекта

- выводы grep/sed с контроллера;
- журнал systemd wb-mqtt-dali;
- наблюдаемые кадры 0x3800/0x38FE;
- A/B-тест при остановленном и запущенном сервисе;
- повторный полевой тест после внесения return в _schedule_confirm().

11.3. Оценка уверенности

Параметр	Значение
Связь проблемы с активностью wb-mqtt-dali	Очень высокая: воспроизводимый A/B-тест.
Связь улучшения с отключением confirmation polls	Очень высокая: точечная правка и успешный повторный тест.
Точный физический механизм пропуска каждого кадра	Средняя: вероятны коллизии/занятость шины, но независимого trace не было.
Рекомендация разделять внешние и собственные команды	Высокая: напрямую следует из архитектуры и устраняет необходимость глобального отключения подтверждений.

Формулировка для тикета

Просьба рассмотреть возможность не запускать немедленный confirmation poll после команд, полученных из bus monitor, либо сделать это поведение настраиваемым/отложенным. На крупной инсталляции wb-mqtt-dali 1.19.4 связанные QueryActualLevel и DT7 QuerySwitchStatus приводили к нестабильности автономных DA-4S; локальное отключение _schedule_confirm() полностью устранило проблему.

Конец отчёта