

Адаптация Telegram-плагина под VK

????????? ???????	Vk
??? ???????	Common.Integrations.Vk
??????????????	?????????????? ??????? ?? Metabot
???????	Metabot
???? ?????????	09 ??????? 2026
?????????? ??? ???????????	09 ??????? 2026

Описание

Плагин Vk позволяет использовать один и тот же скрипт бота в Telegram, Max и VK. Он не вводит новый класс: плагин «патчит» методы класса TelegramMessage из плагина Common.Integrations.Telegram.

- Если лид пришёл из Telegram — вызывается исходный метод Telegram (или патч Max, если он подключён раньше).
- Если лид пришёл из Max — срабатывает Max-патч.
- Если лид пришёл из VK — те же самые методы (send, edit, addReplyToText, removeInlineKeyboard, removeReplyKeyboard, getMessagePayload, debug) отправляют запросы в VK API.

Плагин поддерживает отправку текстовых сообщений, создание inline-клавиатур (кнопки text и open_link), обработку нажатий и текстового ввода, а также отправку медиагрупп (по одному вложению). PHP-драйвер VK не меняется: часть поведения Telegram/Max в VK осознанно деградирует (см. таблицу совместимости).

Документация VK API: <https://dev.vk.com/method/messages.send>

Подключение

Сначала подключается Telegram-плагин, затем при необходимости Max, затем — Vk. Порядок важен: Vk патчит уже загруженный класс. Повторное подключение безопасно (патч ставится один раз).

```
let TelegramMessage = require('Common.Integrations.Telegram')
require('Common.Integrations.Max') // опционально
require('Common.Integrations.Vk')

let msg = new TelegramMessage()
```

Больше ничего в скрипте менять не нужно: проверять канал (if (isVk)) не требуется.

Основные параметры

```
msg.text = "Ваше сообщение" // Текст сообщения
msg.parse_mode = "HTML" // В VK разметка снимается до plain text
msg.keyboard = "Да[ yes]==Нет[ no]" // Создание клавиатуры
msg.entities = [] // В VK игнорируются (текст очищается так же, как при parse_mode)
```

Параметр	Telegram	Max	VK
<code>text</code>	<code>text</code>	<code>text</code>	<code>message</code> (plain)
<code>parse_mode</code>	<code>parse_mode</code>	<code>format</code> (HTML → html, Markdown / MarkdownV2 → markdown)	strip → plain
<code>keyboard</code>	inline-клавиатура	вложение <code>inline_keyboard</code>	VK <code>keyboard</code> JSON (<code>inline: true</code>)
<code>entities</code>	<code>entities</code>	преобразуются в HTML	strip
<code>protect_content</code>	поддерживается	игнорируется	игнорируется

Форматы клавиатуры

Формат строки клавиатуры полностью совпадает с Telegram-плагином.

- Вертикальное разделение — используйте ==;

```
msg.keyboard = "Кнопка1[ btn1]==Кнопка2[ btn2]" // Кнопки будут расположены вертикально
```

- Горизонтальное разделение — используйте ||;

```
msg.keyboard = "Кнопка1[ btn1] || Кнопка2[ btn2] " // Кнопки будут расположены горизонтально
```

- Комбинированное разделение;

```
msg.keyboard = "Кнопка1[ btn1] || Кнопка2[ btn2] ==Кнопка3[ btn3] || Кнопка4[ btn4] "
```

Значение в квадратных скобках для обычной кнопки отправляется в VK как `action.type=text` с `payload` в виде JSON-строки `"btn_static_<значение>"` (VK требует валидный JSON, иначе ошибка API 911). В `getMessagePayload().callback_data` кавычки и префикс снимаются — так же, как в Telegram/Max.

Специальные типы кнопок

- Запрос контакта (`request_contact`) в VK **не поддерживается**: кнопка пропускается, пишется отладочное сообщение:

```
msg.keyboard = "Отправить контакт[ telegram_contact] "
```

- Запрос локации (`request_geo_location`) в VK **не поддерживается**: кнопка пропускается, пишется отладочное сообщение:

```
msg.keyboard = "Отправить локацию[ telegram_location] "
```

- Ссылки (`open_link`):

```
msg.keyboard = "Посетить сайт[ https: //example. com] "
```

- Web App. Запись `{https://...}` в VK **не поддерживается**: кнопка пропускается, пишется отладочное сообщение:

```
msg.keyboard = "Открыть приложение{https: //webapp-url. com}"
```

- Ссылка на пользователя `tg://user?id=...` в VK не поддерживается: кнопка пропускается, пишется отладочное сообщение.
- Reply-клавиатура (`msg.reply_markup = 'keyboard'`) в VK эмулируется теми же inline-кнопками; нажатие с `payload` при `_reply_markup == 'keyboard'` в `getMessagePayload()` даёт `input_type: 'write'`.

Лимит подписи кнопки VK — **40 символов** (плагин обрезает).

Примеры использования

Простое меню с обработкой ответов

Код будет работать корректно только в команде JS Callback. Один и тот же код работает для Telegram, Max и VK.

```
let TelegramMessage = require('Common.Integrations.Telegram')
require('Common.Integrations.Max')
require('Common.Integrations.Vk')

let msg = new TelegramMessage()
msg.text = "Выберите действие: "
msg.keyboard = "Информация[ info]==Помощь[ help]==Настройки[ settings]"
msg.parse_mode = "HTML"

switch (true) {
case (isFirstImmediateCall):
    msg.send()
    return false

case (msg.getMessagePayload()?.callback_data == "info"):
    msg.addReplyToText()
    bot.sendMessage("Информация о боте...")
    return false

case (msg.getMessagePayload()?.callback_data == "help"):
    msg.addReplyToText()
    bot.sendMessage("Справка по использованию...")
    return false

case (msg.getMessagePayload()?.callback_data == "settings"):
    msg.addReplyToText()
    bot.runScriptForLead(123, leadId) // Запуск скрипта настроек
    return false
}
```

Минимальное демо: кнопки + переход дальше

```
const TelegramMessage = require('Common.Integrations.Telegram')
require('Common.Integrations.Max')
require('Common.Integrations.Vk')
```

```
const msg = new TelegramMessage()  
msg.text = 'VK+Telegram+Max demo\n\nВыберите кнопку:'  
msg.keyboard = 'Кнопка A[ a]==Кнопка B[ b]'  
msg.parse_mode = 'HTML'  
  
if (isFirstImmediateCall) {  
    msg.send()  
    return false  
}  
  
msg.addReplyToText()  
return true
```

В Telegram и Max после нажатия клавиатура исчезнет, а к тексту сообщения добавится строка ► Кнопка A. В VK подпись кнопки уже приходит в чат как сообщение пользователя (это text-кнопки), поэтому addReplyToText() ничего не отправляет — только очищает _keyboard и возвращает getMessagePayload().

Справочник методов

Основные методы отправки

- send() — отправка нового сообщения (messages.send). ID сохраняется в атрибуты лида vk_last_message_id и telegram_last_message_id. Клавиатура уходит напрямую в теле messages.send (драйвер VK не передаёт keyboard из sendMessage);
- edit() — в VK нет редактирования сообщений бота, поэтому отправляется новое сообщение (как send);=
- addReplyToText() — в VK no-op: сообщение не отправляется (выбор уже виден как это подписи кнопки). Возвращает getMessagePayload() и очищает _keyboard;
- removeInlineKeyboard() — в VK нельзя убрать кнопки из уже отправленного сообщения, поэтому метод только очищает _keyboard и возвращает getMessagePayload();
- removeReplyKeyboard() — отправляет текст _remove_keyboard_text через messages.send без клавиатуры (как уведомление в Telegram).

Получение информации

- getMessagePayload() — получение информации о сообщении из вебхука:
 - message_id — ID сообщения VK;
 - text — текст сообщения / подпись кнопки;
 - input_type — тип ввода ('write' / 'press');

- `callback_data` — данные колбэка (без префикса `btn_static_`);
- `original_reply_markup` — какой тип клавиатуры хотели получить (`inline_keyboard` / `keyboard`);
- `payload` — полные данные вебхука.

Обработка ответов пользователя:

```
// Проверка типа ввода
if (msg.getMessagePayload()?.input_type == "write") {
    // Пользователь написал текст
}

// Проверка нажатия кнопки
if (msg.getMessagePayload()?.callback_data == "button_id") {
    // Пользователь нажал кнопку
}
```

Структура вебхука VK отличается от Telegram: событие всегда `message_new`. Сообщение лежит в `object.message` (API ≥ 5.103) или прямо в `object`. Нажатие text-кнопки приходит как `message_new` с полем `payload` (JSON-строка); плагин декодирует его, ищет кнопку в `_keyboard` и выставляет `input_type=press`. Текст без `payload` — `input_type=write`; если подпись совпала с кнопкой текущей клавиатуры, `callback_data` берётся из неё.

Отладка

```
msg.debug("Отладочное сообщение")
```

В Telegram при `bot_debug_level == 0` используется «тихий» `getMe`. В VK такого метода нет: при уровне 0 отладка не отправляется пользователю. При ненулевом уровне сообщение уходит в чат (как и в Telegram-плагине).

Лучшие практики

- Всегда проверяйте первый вызов:

```
if (isFirstImmediateCall) {
    msg.send()
    return false
}
```

- Добавляйте обработку текстовых сообщений:

```
if (msg.getMessagePayload()?.input_type == "write") {  
    bot.sendMessage('Пожалуйста, используйте кнопки')  
    return false  
}
```

- После нажатия кнопки вызывайте `addReplyToText()` — в Telegram/Max это скроет клавиатуру и допишет ► ..., в VK вызов безопасен (no-op) и сохраняет единый код скрипта:

```
msg.addReplyToText()
```

- Подключайте `Common.Integrations.Vk` сразу после Telegram (и Max, если он нужен) в начале скрипта.
- Не используйте в тексте кнопок больше 40 символов; payload кнопки VK — до 255 байт.
- Для кросс-канальных скриптов всё равно задавайте `parse_mode = 'HTML'`: в Telegram/Max форматирование сохранится, в VK плагин снимет разметку сам.
- В настройках сообщества VK включите «Возможности ботов» — иначе `messages.send` с клавиатурой вернёт ошибку 912.

Форматирование текста и entities

VK не принимает HTML/Markdown в `messages.send` так же, как Telegram/Max. Если заданы `parse_mode` или `entities`, плагин удаляет HTML-теги, типичные `entities` и маркеры Markdown и отправляет plain text.

```
msg.text = "<b>Жирный текст</b>\n<i>Курсив</i>\n<code>Моноширинный текст</code>"  
msg.parse_mode = "HTML"  
// В VK уйдёт: "Жирный текст\nКурсив\nМоноширинный текст"
```

Работа с entities

`msg.entities` в VK не отправляются. Текст очищается так же, как при `parse_mode`.

```
let TelegramMessage = require('Common.Integrations.Telegram')  
require('Common.Integrations.Max')  
require('Common.Integrations.Vk')  
  
let msg = new TelegramMessage()  
let messageData = lead.getJsonAttr("messageData")
```

```
msg.text = messageData?.text
msg.entities = messageData?.entities

// Не используйте одновременно parse_mode и entities
```

Отправка медиагрупп

Глобальная функция `sendMediaGroup` для лидов VK отправляет вложения по одному через `bot.sendMessage` с `{ type, url }`. Типы элементов приводятся так: `photo/image` → `image`, `document/file` → `file`, `video` → `video`, `audio` → `audio`. Подпись очищается от разметки. Для Telegram/Max-лидов вызывается исходная (или Max) реализация.

Работа с файлами

Входящие файлы обрабатываются стандартными методами бота — так же, как в Telegram-плагине. Вы можете получить URL файла и сохранить его в карточку лида.

```
let TelegramMessage = require('Common.Integrations.Telegram')
require('Common.Integrations.Max')
require('Common.Integrations.Vk')
let msg = new TelegramMessage()

// Все входящие файлы
let attachments = bot.getAllAttachments()

if (Boolean(attachments?.[0]?.url)) {
  let uploadData = bot.downloadFileFromUrl(attachments[0].url)

  msg.debug('Пользователь отправил файл: ' + uploadData.url)
  lead.setAttr('file', uploadData.url)

  // Запуск скрипта обработки
  bot.runScriptByCodeForLead("recieveFile", lead.getData('id'))

  bot.stop()
}
```

Пример ожидания фото:

```

let TelegramMessage = require('Common.Integrations.Telegram')
require('Common.Integrations.Max')
require('Common.Integrations.Vk')
let msg = new TelegramMessage()

msg.text = 'А теперь жду твоё фото '
msg.parse_mode = 'HTML'

let data = bot.getAllAttachments()

switch (true) {
case (isFirstImmediateCall):
    msg.send()
    return false

case (Boolean(data?.[0]?.type !== 'image')):
    bot.sendMessage('Отправь изображение. Такой формат не подходит')
    return false

case (Boolean(data?.[0]?.type === 'image')):
    let uploadData = bot.downloadFileFromUrl(data[0].url)
    msg.debug('Пользователь отправил файл: ' + uploadData.url)
    lead.setAttr('photo', uploadData.url)
    bot.sendMessage('Класс! Фото загрузил')
    return true

default:
    msg.addReplyToText()
    return true
}

```

Рекомендации по работе с файлами

- Всегда проверяйте наличие вложения (attachments?.[0]?.url) перед скачиванием.
- Сохраняйте URL, который вернул bot.downloadFileFromUrl(), а не исходный URL канала — он может устареть.
- Не используйте Telegram file_id для VK: идентификаторы вложений в каналах разные.

Совместимость с Telegram-плагином

Возможность	Telegram	Max	VK	Комментарий
<code>send()</code>	□	□	□	VK: <code>messages.send</code>
<code>edit()</code>	□	□	△	VK: новое сообщение
<code>addReplyToText()</code>	□	□	△	VK: ничего не отправляет (эхо кнопки уже в чате)
<code>removeInlineKeyboard()</code>	□	□	△	VK: ничего не отправляет
<code>removeReplyKeyboard()</code>	□	△	△	VK: <code>send</code> <code>remove_keyboard_text</code>
<code>getMessagePayload()</code>	□	□	□	
<code>debug()</code>	□	□	△	при <code>bot_debug_level == 0</code> — silent no-op
<code>parse_mode = HTML</code>	□	□	□	strip → plain
<code>parse_mode = MarkdownV2</code>	□	△	□	strip → plain
<code>entities</code>	□	△	□	strip
Кнопки contact / location	□	□	□	кнопка пропускается + debug
Кнопки-ссылки	□	□	□	<code>open_link</code>
Web App <code>{url}</code>	□	△	□	кнопка пропускается + debug
<code>tg: //user?id=</code>	□	□	□	кнопка пропускается
<code>protect_content</code>	□	□	□	игнорируется
<code>TelegramPhoto</code>	□	□	□	не патчится, используйте <code>sendMediaGroup</code>
<code>sendMediaGroup</code>	□	□	△	по одному вложению
Эхо подписи кнопки	нет	нет	да	особенность VK text-кнопок

Ограничения VK

- Текст кнопки — до 40 символов, payload кнопки — до 255 байт.
- payload должен быть валидной JSON-строкой (плагин кодирует сам); иначе ошибка 911.

- Для клавиатур в сообществе должны быть включены «Возможности ботов»; иначе ошибка 912.
 - Нет in-place edit / снятия inline-клавиатуры без правок PHP-драйвера.
 - Нет message_event (callback без эха подписи) без правок PHP-драйвера.
 - Вебхук Callback API требует публичный HTTPS URL (для локальной разработки — туннель; APP_URL может оставаться на localhost).
-

Версия #2

Ирина Петрова создал 26 August 2026 10:44:00

Ирина Петрова обновил 21 October 2026 10:54:26