Адаптация Telegram-плагина под MAX

???????? ???????	Common.Integrations.Max
???????????	??????????? ??????? ?? Metabot
??????	Metabot
???? ????????	05 ??????? 2026
????????? ???? ??????????	05 ??????? 2026

Описание

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

- Если лид пришёл из **Telegram** — работают оригинальные методы Telegram-плагина (ничего не меняется).
- Если лид пришёл из **Max** — те же самые методы (`send`, edit, addReplyToText, removeInlineKeyboard, removeReplyKeyboard, getMessagePayload, `debug`) отправляют запросы в Max Bot API.

Плагин поддерживает отправку текстовых сообщений, создание inline-клавиатур, обработку нажатий кнопок и ответов пользователя, редактирование сообщений и отправку медиагрупп.

Документация Max Bot API: https://dev.max.ru/docs-api

Адаптация плагина в коде

Для работы плагина Telegram в MAX, после подключения основного класса TelegramMessage добавьте вызов сниппета **Business.Integrations.MAX**. Затем добавьте инициализацию плагина.

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

```
let TelegramMessage = require('Common.Integrations.Telegram')
snippet('Business.Integrations.MAX')

let msg = new TelegramMessage()
```

По аналогии адаптируется плагин TelegramMedia.

```
let { TelegramMediaGroup } = require('Common.Integrations.TelegramMedia')
snippet('Business.Integrations.MAXmedia')

let mediaMsg = new TelegramMediaGroup()
```

Основные параметры

```
msg.text = "Ваше сообщение" // Текст сообщения
msg.parse_mode = "HTML" // Режим форматирования (HTML или Markdown)
msg.keyboard = "Да[yes]==Нет[no]" // Создание клавиатуры
msg.entities = [] // Telegram entities (в Max автоматически преобразуются в HTML)
```

Параметр	Telegram	Max
`text`	`text`	`text`
`parse_mode`	`parse_mode`	`format` (`HTML` → `html`, `Markdown`/`MarkdownV2` → `markdown`)
`keyboard`	inline-клавиатура	вложение `inline_keyboard`
`entities`	`entities`	преобразуются в HTML-теги и отправляются с `format: html`
`protect_content`	поддерживается	игнорируется

Специальные типы кнопок

Ссылки (`link`):

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

Web App. В Max кнопка `open_app` требует username бота, поэтому запись `{https://...}` превращается в обычную кнопку-ссылку (`link`):

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

Ссылка на пользователя `tg://user?id=...` в Max **не поддерживается**: кнопка пропускается, в карточку лида пишется отладочное сообщение.

Справочник методов

Основные методы отправки

- send() — отправка нового сообщения (`POST /messages`). ID отправленного сообщения сохраняется в атрибут лида max_last_message_id;
- edit() — редактирование существующего сообщения (`PUT /messages?message_id=`): меняются текст, формат и клавиатура;
- addReplyToText() — после нажатия кнопки убирает клавиатуру и добавляет к тексту сообщения ➤ <текст нажатой кнопки>. Для текстового ввода (`write`) только убирает клавиатуру;
- removeInlineKeyboard() — удаление клавиатуры. К тексту сообщения добавляется ➤ □□;
- removeReplyKeyboard() — в Max нет reply-клавиатур, поэтому метод убирает кнопки с последнего сообщения и отправляет текст _remove_keyboard_text, если он задан.

В Max клавиатура — это вложение сообщения. При редактировании остальные вложения исходного сообщения сохраняются, меняется только inline_keyboard.

Получение информации

- getMessagePayload() — получение информации о сообщении из вебхука:
- message_id — ID сообщения (`mid`). Для нажатия кнопки — mid сообщения с клавиатурой;
- text — текст сообщения/кнопки;
- input_type — тип ввода (`'write'` / `'press'`);
- callback_data — данные колбэка (без префикса `btn_static_`);
- callback_id — идентификатор колбэка Max (только для нажатия кнопки);
- original_reply_markup — клавиатура исходного сообщения;
- 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 Max отличается от Telegram: для нажатия кнопки (`message_callback`) сообщение с клавиатурой лежит в payload.message, а данные нажатия — в payload.callback. Для текста (`message_created`) текст находится в payload.message.body.text. Плагин приводит это к единому виду getMessagePayload().

Форматирование текста и entities

> Формат строки клавиатуры полностью совпадает с Telegram-плагином.

Max принимает форматирование через параметр format: html или markdown.
HTML форматирование

Поддерживаются теги <b>, <i>, <u>, <s>, <code>, <pre>, <blockquote>, <a href="...">.

```
msg.text = "<b>Жирный текст</b>\n<i>Курсив</i>\n<code>Моноширинный текст</code>"
msg.parse_mode = "HTML"
```

Markdown форматирование

parse_mode со значениями Markdown, MarkdownV2 и md отправляется в Max как format: markdown. Синтаксис Markdown в Max не совпадает с MarkdownV2 Telegram (в частности, экранирование), поэтому для кросс-канальных скриптов используйте HTML.
Работа с entities

В Telegram entities позволяют сохранить форматирование, которое прислал пользователь. В Max их отправить нельзя, поэтому плагин автоматически преобразует msg.entities в HTML и отправляет сообщение с format: html. Поддерживаются типы: bold, italic, underline, strikethrough, code, pre, blockquote, text_link. Остальные типы пропускаются, спецсимволы текста экранируются.

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

let msg = new TelegramMessage()
```

```
let messageData = lead.getJsonAttr("messageData")

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

// Не используйте одновременно parse_mode и entities
```

> Важно: входящее форматирование Max приходит в message.body.markup, а не в entities, поэтому «эхо» форматирования пользователя из Max через entities без дополнительной конвертации не работает.
> Отправка медиагрупп

Глобальная функция sendMediaGroup для лидов Max отправляет вложения в Max. Типы элементов приводятся так: photo`/`image → image, document`/`file → file, video → video, audio → audio. Для Telegram-лидов работает оригинальная реализация.

Рекомендации по работе с файлами

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

> Входящие файлы обрабатываются стандартными методами бота — так же, как в Telegram-плагине.

Совместимость с Telegram-плагином

Возможность	Telegram	Max	Комментарий
`send()`	✅	✅	
`edit()`	✅	✅	Меняет текст, формат и клавиатуру
`addReplyToText()`	✅	✅	Добавляет `➤ <кнопка>`, убирает клавиатуру

`removeInlineKeyboard()`	✅	✅	Добавляет `➤ ✅`
`removeReplyKeyboard()`	✅	⚠️	Reply-клавиатур в Max нет, эмулируется через inline-кнопки
`getMessagePayload()`	✅	✅	Дополнительно возвращает `callback_id`
`debug()`	✅	✅	
`parse_mode = HTML`	✅	✅	
`parse_mode = MarkdownV2`	✅	⚠️	Синтаксис Max Markdown отличается
`entities`	✅	⚠️	Конвертируются в HTML на отправке
Кнопки contact / location	✅	✅	`request_contact` / `request_geo_location`
Кнопки-ссылки	✅	✅	
Web App `{url}`	✅	⚠️	Заменяется на кнопку-ссылку
`tg://user?id=`	✅	✅	Кнопка пропускается
`protect_content`	✅	✅	Игнорируется
`TelegramPhoto`	✅	✅	Не патчится, используйте `sendMediaGroup`
`sendMediaGroup`	✅	✅	

Ограничения Max

- Текст кнопки — до 128 символов, payload кнопки — до 1024 символов.
- Ссылки на локальные адреса (`http://127.0.0.1`) не принимаются при подключении вебхука Max — нужен публичный HTTPS-адрес.

- Нажатие кнопки приходит как message_callback; для текстового ввода — message_created.

> Подробнее примеры использования, форматы клавиатуры и лучшие практики смотрите в **Документации Telegram-плагина**

Версия #2
Ирина Петрова создал 2026-09-23 08:19:38 UTC
Ирина Петрова обновил 2026-10-08 12:13:01 UTC