Интеграция плеера

Базовая интеграция

Чтобы встроить видео с RUTUBE на сторонний сайт, нужно добавить код вставки плеера в код страницы, на которую хотите встроить плеер. Встроить можно любое видео, которое не скрыто автором. Видео, доступное только по прямой ссылке, тоже можно встроить — для этого понадобится доступ в студию RUTUBE автора этого видео.

1. Найдите нужное видео на сайте RUTUBE и нажмите «Поделиться» → «Код вставки плеера»
2. Скопируйте код. Чтобы это сделать, кликните на код или на кнопку «Скопировать»
3. Вставьте код вставки плеера в код страницы, на которую хотите встроить видео

Пример:

```
<iframe
         width="720"
         height="405"
         src="https://rutube.ru/play/embed/3333514097a5241aff11e075b6b22325"
         frameBorder="0"
         allow="clipboard-write; autoplay"
         webkitAllowFullScreen
         mozallowfullscreen
         allowFullScreen
 ></iframe>
```

Доступные query-параметры

- `skinColor` **** — Hex без решетки. Изменяет цвет некоторых элементов в интерфейсе
- `t` **** — время в секундах, с которого начнётся воспроизведение
- `stopTime` **** — время в секундах, на котором воспроизведение завершится
- `q(quality)` — параметр для настройки начального качества видео. Доступные значения: `144, 240, 360, 480, 720, 1080, 1440, 2160, auto` Если у пользователя имеется сохраненное качество видео от предидущих просмотров, именно оно будет выбрано для инициализации
- autoplay={true|false} — переопределяет значение параметра auto_start настроенного на сервере
- naked={true|false} — скрывает все контролы в плеере, в том числе "пропустить" для рекламы.
- autostartmute={true|false} — видео стартует без звука/со звуком

Пример:

```
<iframe
         width="720"
         height="405"
         src="https://rutube.ru/play/embed/3333514097a5241aff11e075b6b22325?
skinColor=e53935&t=300&stopTime=480&q=1080"
         frameBorder="0"
         allow="clipboard-write; autoplay"
         webkitAllowFullScreen
         mozallowfullscreen
         allowFullScreen
 ></iframe>
```

Инициализация плеера

Получаемые события от плеера через postMessage

player:ready

Описание

Отправляется после инициализации провайдера.

Для взаимодействия с плеером необходимо дождаться этого эвента

Данные

```
 {
    "type": "player:ready",
    "data": {
        "clientId": "8491de01-0766-4e0c-96a6-f82252aacffc", //  UUID
        "videoId": "57e1cb24939d657284dc2900deb5949a", //  video
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:ready') {
      console.log('player:ready data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:init

Описание

Отправляется после того как загружены данные для воспроизведения и выбран тип провайдера

Данные

```
 {
    "type": "player:init",
    "data": {
        "clientId": "33cc0272-6897-4b28-85e3-349505c4197b", //  UUID
        "videoId": "3deaff3ef3225202d858cda734f8e55f", //  video
        "playerId": "video_frame" //   (    ;   )
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:init') {
      console.log('player:init data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:playOptionsLoaded

Описание

Отправляется при наличии в query параметра `getPlayOptions` с перечислением ключей, которые хотим получить из запроса `/api/play/options/57e1cb24939d657284dc2900deb5949a/`

Например:

`?getPlayOptions=id,thumbnail_url`

Данные

```
 {
    "type": "player:playOptionsLoaded",
    "data": {
                "playOptions": {
                    "thumbnail_url": "https://pic.rutube.ru/video/a5/b9/a5b96f6da553beb3d2b474e5933f4da8.jpg",
                           "title": "  (, .  , 1978 .)",
                    "video_id": "e88997f2e6e8d9f134f77a3fe5cc3c8d",
                    "video_url": "https://rutube.ru/video/e88997f2e6e8d9f134f77a3fe5cc3c8d/",
                },
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:playOptionsLoaded') {
      console.log('player:playOptionsLoaded data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Управление контролами плеера

Получаемые события от плеера через postMessage

player:controlsVisibilityChanged

Описание

Событие отправляется, при показе/скрытии всех контроллов.

Параметры

Пример:

```
{
        type: 'player:controlsVisibilityChanged',
        data: {
                playerId: "video_frame"
                visible: true//  true -  , false -
        }
}
```

Отправляемые события в плеер через postMessage

player:showControls

player:hideControls

Описание

Позволяет отображать или скрывать все контроллы в плеере.

player:showControls

Показывать нижние контроллы в плеере

Данные

```
 window.frames.postMessage(JSON.stringify(
{ type: 'player:showControls' }
));
```

player:hideControls

Скрывать нижние контроллы в плеере

Данные

```
window.frames.postMessage(JSON.stringify(
{ type: 'player:hideControls' }
));
```

player:enterNakedMode

Скрывает все контроллы в плеер, включая заголовок

Данные

```
window.frames.postMessage(JSON.stringify(
{ type: 'player:enterNakedMode' }
));
```

player:exitNakedMode

Выход из naked режима.

Данные

```
window.frames.postMessage(JSON.stringify(
{ type: 'player:exitNakedMode' }
));
```

Проигрывание

Получаемые события от плеера через postMessage

player:playStart

Описание

Первый старт проигрывания

Данные

```
 {
    "type": "player:playStart",
    "data": {
        "video_id": "e30edf24ea097b45d0d469fac449478c", //
        "qa": true, //
        "sm": "default", //   default/cinema
        "yclid": "1715854197193969726", // global client id
        "viewid": "339f02f2c72f0d0f2676c1ec6c069d3b", // REQUEST_ID    ,     ,     UUID4
        "user": null, //  ,
        "cid": "d65fba3c-75a7-450d-b273-4e86b0569791", // uuid?
        "pid": "c97972a0-8269-402a-82e3-0192366dc98b", // guid?
        "referer": "http%3A%2F%2Flocalhost%3A8080", //
        "v": 100, //
        "q_w": 1920, //
        "q_h": 1080, //
        "tr": "hls", //
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:playStart') {
      console.log('player:playStart data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:playComplete

Описание

Срабатывает по окончанию видео

Данные

```
 {
    "type": "player:playComplete",
    "data": {
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:playComplete') {
      console.log('player:playComplete data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:changeState

Описание

Изменение состояния плеера

Параметры

- `state: string -`
    - `:`

      `initalazing, playing, pause, advert, change, completed, buffering, error, loading, seeking, seeked`
- `status: string - ,  state`
    - `:`

      `initalazing, playing, pause, advert, change, completed, buffering, error, loading, seeking, seeked`
- `isLicensed: boolean -`
- `phase: string -`
    - `:`

      `prepare, playing, complete`
- `reason: string -`
    - `:`

      `cuepoint, postMessage, next`
- `playerId: string`

Данные

```
 {
    "type": "player:changeState",
    "data": {
        "state": "playing",
                "status": "playing",
        "isLicensed": false,
                "phase": "playing",
                "reason": "",
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:changeState') {
      console.log('player:changeState data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:play

Описание

Воспроизведение видео

Для корректного воспроизведения, отправлять это событие стоит после готовности плеера (событие `player:ready`)

Параметры

Пример:

```
{type: 'player:play'}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:play'}), '*');
```

player:pause

Описание

Постановка видео на паузу

Для корректной постановки на паузу, отправлять это событие стоит после готовности плеера (событие `player:ready`)

Параметры

Пример:

```
{type: 'player:pause'}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:pause'}), '*');
```

player:stop

Описание

Остановка видео

Для корректной постановки на паузу, отправлять это событие стоит после готовности плеера (событие `player:ready`)

Параметры

Пример:

```
{type: 'player:stop'}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:stop'}), '*');
```

player:suspend

Описание

Остановка видео и прекращение загрузки.

Не безопасный метод. Используется в сочетании с `player:wakeup` **Параметры**

Пример:

```
{type: 'player:suspend'}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:suspend'}), '*');
```

player:wakeup

Описание

Воспроизведение видео и восстановление загрузки.

Обычно используется после `player:suspend` , для восстановления загрузки

Параметры

Пример:

```
{type: 'player:wakeup'}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:wakeup'}), '*');
```

Управление прогрессом

Получаемые события от плеера через postMessage

player:currentTime

Описание

Срабатывает при каждом изменении текущего времени воспроизведения.

Параметры

- `time: number -`
- `currentTime: number -      ( ,   time)`
- `duration: number -`

Данные

```
{
    "type": "player:currentTime",
    data: {
                time: 134.834334,
            currentTime: 134.834334,
                duration: 1000
        }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:currentTime') {
      console.log('player:currentTime data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:durationChange

Описание

Отправляется при изменении длительности видео (например при стриминге).

Данные

```
{
    "type": "player:durationChange",
    "data": {
        "duration": 4816.68, //
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:durationChange') {
      console.log('player:durationChange data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:setCurrentTime

Описание

Установка текущего времени воспроизведения

Параметры

- time: number - время в секундах

Пример:

```
{
  type: 'player:setCurrentTime',
  data: {
          time: 300 //
  }
}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({
  type: 'player:setCurrentTime',
  data: {
          time: 0 //
  }
}), '*');
```

player:relativelySeek

Описание

Событие для перемотки видео

Параметры

- time: number - время в секундах

Пример:

```
{
        type: 'player:relativelySeek',
        data: {
          time: 20 //
        }
}
```

Пример отправки события

```
const iframe = document.getElementById('player');
//
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:relativelySeek', data: {
    time: 20 //
  }}),
  '*'
);

//
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:relativelySeek', data: {
          time: -20 //
  }}),
  '*'
);
```

Звук

Получаемые события от плеера через postMessage

player:volumeChange

Описание

Отправляется при включении/отключении звука и при изменении уровня громкости

Данные

```
{
    "type": "player:volumeChange",
    "data": {
        "volume": "100.00", //
        "muted": false, //
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:volumeChange') {
      console.log('player:volumeChange data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:mute

Описание

Событие для отключения звука

Параметры

Пример:

```
{type: 'player:mute'}
```

Пример прослушивания

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:mute'}), '*');
```

player:unMute

Описание

Событие для включения звука

Параметры

Пример:

```
{type: 'player:unMute'}
```

Пример прослушивания

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(JSON.stringify({type: 'player:unMute'}), '*');
```

player:setVolume

Описание

Событие для установки уровня громкости

Параметры

- volume: number - устанавливает текущий уровень громкости от 0 до 1
- change: number - величина изменения уровня громкости от -1 до 1. Отрицательные значения используются для понижения уровня громкости

Пример:

```
//
{
        type: 'player:setVolume',
        data: {
                volume: 0.2
        }
}

//
{
        type: 'player:setVolume',
        data: {
                change: 0.1
        }
}
```

Пример прослушивания

```
const iframe = document.getElementById('player');
//
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:setVolume', data: {
          volume: 0.2 //   ( 0  1)
  }}),
  '*'
);

//
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:setVolume', data: {
          change: 0.1 //
  }}),
  '*'
);

//
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:setVolume', data: {
          change: -0.1 //
  }}),
  '*'
);
```

Управление качеством

Получаемые события от плеера через postMessage

player:qualityList

Описание

Отправляется после загрузки уровней качества видео

Данные

```
{
    "type": "player:qualityList",
    "data": {
        "list": [144, 240, 360, 480, 720, 1080, 1440, 2160], //
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:qualityList') {
      console.log('player:qualityList data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:currentQuality

Описание

Отправляется при изменении текущего уровня качества видео

Данные

```
{
    "type": "player:currentQuality",
    "data": {
        "quality": {
            "height": 1080, //
            "quality": "1080", //
            "isAutoQuality": false //
        },
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:currentQuality') {
      console.log('player:currentQuality data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:changeQuality

Описание

С помощью этого события можно изменить текущее качество видео

Для корректного изменения, отправлять это событие стоит после загрузки списка качеств у видео (событие `player:qualityList`)

Параметры

- `quality: string` - уровень качества
    - Доступные значения

`'144', '240', '360', '480', '720', '1080', '1440', '2160', 'auto'`

Пример:

```
{
        type: 'player:changeQuality',
        data: {
                quality: '1080' //
        }
}
```

Пример отправки события

```
const playerIframe = document.getElementById('player');
playerIframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:changeQuality', data: {quality: '1080'}}),
  '*'
);
```

Управление плеером

Получаемые события от плеера через postMessage

player:changeFullscreen

Описание

Отрабатывает при включении/отключении полноэкранного режима

Данные

```
{
    "type": "player:changeFullscreen",
    "data": {
        "isFullscreen": true, //
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:changeFullscreen') {
      console.log('player:changeFullscreen data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:setSkinColor

Описание

Изменение цвета некоторых элементов плеера

Параметры

- color: string - Hex без решетки.

Пример:

```
{
        type: 'player:setSkinColor',
        data: {
          color: 'cd5c5c'
        }
}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(
        JSON.stringify({type: 'player:setSkinColor', data: {
          color: 'cd5c5c'
        }}),
        '*'
);
```

player:enterFullscreen

Описание

Переход в полноэкранный режим, если возможно

Пример:

```
{
        type: 'player:enterFullscreen',
}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(
        JSON.stringify({type: 'player:enterFullscreen'}),
        '*'
);
```

player:exitFullscreen

Описание

Переход в полноэкранный режим, если возможно

Пример:

```
{
        type: 'player:exitFullscreen',
}
```

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(
        JSON.stringify({type: 'player:exitFullscreen'}),
        '*'
);
```

Управление видео

Получаемые события от плеера через postMessage

player:buffering

Описание

Отправляется во время буфферизации

Данные

```
{
    "type": "player:buffering",
    "data": {
        "range": [
            {
                "start": 0,
                "end": 0
            },
            {
                "start": 0,
                "end": 10
            },
            {
                "start": 150.000907,
                "end": 350
            }
        ], //
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:buffering') {
      console.log('player:buffering data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:changeVideo

Описание

Изменение проигрываемого видео

Параметры

- id: string - идентификатор видео
- params
    - quality: string - уровень качества при инициализации видео Доступные значения: `144, 240, 360, 480, 720, 1080, 1440, 2160, auto`

Пример отправки события

```
const iframe = document.getElementById('player');
iframe.contentWindow.postMessage(
  JSON.stringify({type: 'player:changeVideo', data: {
    id: '9bbdd154783da1bd9b5affe71d8b8331',
                params: {
                        quality: '1080'
                }
  }}),
  '*'
);
```

Пример получения id видеоролика Rutube по id IMDB

Отправляется запрос на ролик с id IMDB

```
GET /api/ratings/imdb/tt1637725/
```

поступает ответ типа ниже с нашим ID видео

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "video_id": "46466621be4dbd99bf344d1cf859335e",
    "imdb_id": "tt1637725"
}
```

Реклама

Получаемые события от плеера через postMessage

player:rollState

Описание

Старт рекламы, закрытие рекламы, успешное окончание рекламы, ошибка в рекламе

Данные

```
{
    "type": "player:rollState",
    "data": {
        "type": "preroll5",  //   *preroll*, *midroll*, *postroll*
                "state": "complete", //     play / complete
        "playerId": "video_frame"
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:rollState') {
      console.log('player:rollState data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:adRequest

Описание

Отправляется после загрузки данных по рекламе

Данные

```
 {
    "type": "player:adRequest",
    "data": {
        "aformat": "preroll4", //   *preroll*, *midroll*, *postroll*
        "did": "5c6182bc6da219ea5c71003b7055cd3b", // fingerprint id
        "videoId": "57e1cb24939d657284dc2900deb5949a", //
        "e": "a_request", //
        "axurl": "<https://rutube.ru/adfoxsdk?
ownerId=240202&p1=bsxkf&p2=fetu&gowast_title=&puid7=0&puid2=22&puid3=28880952>", // URL
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:adRequest') {
      console.log('player:adRequest data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:adStart

Описание

Отправляется после старта рекламы

Данные

```
 {
    "type": "player:adStart",
    "data": {
        "aformat": "preroll4", //   *preroll*, *midroll*, *postroll*
        "did": "5c6182bc6da219ea5c71003b7055cd3b", // fingerprint id
        "videoId": "57e1cb24939d657284dc2900deb5949a", //
        "e": "a_start", //
        "axurl": "<https://rutube.ru/adfoxsdk?
ownerId=240202&p1=bsxkf&p2=fetu&gowast_title=&puid7=0&puid2=22&puid3=28880952>", // URL
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:adStart') {
      console.log('player:adStart data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:adEnd

Описание

Отправляется после успешного завершения рекламы

Данные

```
 {
    "type": "player:adEnd",
    "data": {
        "aformat": "preroll4", //   *preroll*, *midroll*, *postroll*
        "did": "5c6182bc6da219ea5c71003b7055cd3b", // fingerprint id
        "videoId": "57e1cb24939d657284dc2900deb5949a", //
        "e": "a_end", //
        "axurl": "<https://rutube.ru/adfoxsdk?
ownerId=240202&p1=bsxkf&p2=fetu&gowast_title=&puid7=0&puid2=22&puid3=28880952>", // URL
        "playerId": "video_frame"
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:adEnd') {
      console.log('player:adEnd data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:adError

Описание

Отправляется при перехвате ошибок от adsdk и воспроизведении рекламы.

Данные

```
 {
    "type": "player:adError",
    "data": {
            "aformat": "preroll2", //   *preroll*, *midroll*, *postroll*
        "did": "6b6f8e301a48b6e61832140bd625fc47", // fingerprint id
        "videoId": "5551dc706488f4ab74c509b01dfc1b2e", //
            "e": "error", //
            "axurl": "https://yandex.ru/ads/adfox/277740/getCode?p1=crunr&p2=gdol&pke=1&eid2=8:
5551dc706488f4ab74c509b01dfc1b2e&pk=2-
21824&puid20=5551dc706488f4ab74c509b01dfc1b2e&puid21=8&puid22=21824&puid23=4&puid24=2&puid25=2&puid26=7&dl=%%
extUrl%%",
            "errorMessage": "ERROR_MESSAGE", // /
            "playerId": "video_frame"
        }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:adError') {
      console.log('player:adError data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Ошибки

Получаемые события от плеера через postMessage

player:error

Описание

Отправляется при ошибки воспроизведенияtype:

Данные

fatal: true | false

type: 'source_deactivated' | 'payment_required' | 'pending_stream' | 'deleted_by_user' | 'deleted_by_platform_support' | 'blocked_by_copyright' | 'deleted_by_rkn' | 'stream_not_started' | 'stream_finished'

message: string // детальное описание ошибки, например "This video is available for users with paid subscription only"

```
{
    "type": "player:error",
    "data": {
        "playerId": "video_frame",
                "fatal": true,
                "type": "deleted_by_user",
                "message": "This video was deleted by user",
    }
}
```

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:error') {
      console.log('player:error data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Управление субтитрами

Получаемые события от плеера через postMessage

player:playOptionsLoaded

Описание

Отправляется после загрузки playOptions. Содержит информацию о субтитрах в поле captions объекта playOptions

Данные


```
{
    "type": "player:playOptionsLoaded",
    "data": {
        "playOptions": {
                    "captions": [
                        {
                            "sub_id": 56,
                            "format": "srt",
                            "langTitle": "\u0420\u0443\u0441\u0441\u043a\u0438\u0439",
                            "file": "https://pic.rutube.ru/subtitle/21/35/21356b6e0f5772d5be59fae098667544.srt",
                            "is_autogenerated": false,
                            "code": "ru"
                        }
                    ]
                }
    }
}
```


Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:playOptionsLoaded') {
      console.log('player:playOptionsLoaded data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:changeCaption

Описание

Событие отправляется при включение/выключение и выборе субтитров в плеере.

Параметры

- `enabled: boolean - /`
- `current: number - id    playOptions.captions`

Данные

```
{
    "type": "player:changeCaption",
    data: {
                enabled: true,
                current: 56
        }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:changeCaption') {
      console.log('player:changeCaption data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

player:cueChange

Описание

Событие отправляется при появлении/исчезновении активной cue (текстовой метки).

Параметры

- `cue: VTTCueData` – данные активной cue.

Данные

```
{
    "type": "player:cueChange",
    data: {
        align: "center"
         endTime: 791.66
         id: "109"
         line: "auto"
         playerId: "video_frame"
         position: "auto"
         size: 100
         snapToLines: true
         startTime: 788.29
         text: "   ."
         vertical: ""
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:cueChange') {
      console.log('player:cueChange data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:setCaption

Описание

С помощью этого события можно включить/выключить и выбрать субтитры

Для корректного изменения, отправлять это событие стоит после загрузки playOptions

Параметры

- `enabled: boolean - /`
- `current: number - id    playOptions.captions`

Пример:

```
{
        type: 'player:setCaption',
        data: {
                enabled: true,
                current: 56
        }
}
```

Пример отправки события

```
window.frames.postMessage(JSON.stringify(
{ type: 'player:setCaption', data: {enabled: true, current: 56} }
));
```

Управление скоростью воспроизведения

Получаемые события от плеера через postMessage

player:playbackSpeedChanged

Описание

Отправляется при смене скорости воспроизведения в плеере

Данные

```
{
    "type": "player:playbackSpeedChanged",
    "data": {
        "speed": 0.75
    }
}
```

Пример прослушивания

```
window.addEventListener('message', ({data}) => {
  try {
    const parsedData = JSON.parse(data);
    if (parsedData.type === 'player:playbackSpeedChanged') {
      console.log('player:playbackSpeedChanged data', parsedData);
    }
  } catch (error) {
    console.warn(error);
  }
});
```

Отправляемые события в плеер через postMessage

player:setPlaybackSpeed

Описание

С помощью этого события можно сменить скорость воспроизведения в плеере

Для корректного изменения, стоит отправлять корректные значения скорости

Параметры

- `speed: number` -
    - `[0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2]`

Пример:

```
{
        type: 'player:setPlaybackSpeed',
        data: {
                speed: 0.5
        }
}
```

Пример отправки события

```
window.frames.postMessage(JSON.stringify(
{ type: 'player:setPlaybackSpeed', data: {speed: 0.75} }
));
```

События коммуникационного баннера

Получаемые события от плеера через postMessage

player:communicationBannerShow

Описание

Событие открытия коммуникационного баннера

```
{
  "type": "player:communicationBannerShow",
  "instanceId": "123",
  "sender": "player",
  "data": {
    "cid": "",
    "onboardingId": 2010,
    "elements": [
      {
        "elementId": 2172,
        "isShown": true,
        "isFinished": false,
        "actions": [
          {
            "actionId": 6116,
            "isPressed": false
          }
        ]
      }
    ],
    "playerId": "video_frame"
  }
}
```

player:internalBtnBanner

Описание

Клик по кнопке коммуникационного баннера с типом internal, url уходит без домена

```
{
  "type": "player:communicationBannerClick",
  "instanceId": "123",
  "sender": "player",
  "data": {
    "actionId": 6116,
    "cid": "",
    "onboardingId": 2010,
    "elements": [
      {
        "elementId": 2172,
        "isShown": false,
        "isFinished": true,
        "actions": [
          {
            "actionId": 6116,
            "isPressed": true
          }
        ]
      }
    ],
    "url": "video/96680d0b8b72b2cba3c7f3510d385798/?erid=CQH36pWzJqLuYYVAgpcTmGhSRPyyd5VGpNkQSD9shkKNGE",
    "playerId": "video_frame"
  }
}
```

player:communicationBannerClick

Описание

клик по кнопке баннера, в том числе крестик закрытия баннера

```
{
  "type": "player:communicationBannerClick",
  "instanceId": "123",
  "sender": "player",
  "data": {
    "actionId": 6116,
    "cid": "",
    "onboardingId": 2010,
    "elements": [
      {
        "elementId": 2172,
        "isShown": false,
        "isFinished": true,
        "actions": [
          {
            "actionId": 6116,
            "isPressed": true
          }
        ]
      }
    ],
    "url": "https://rutube.ru/video/96680d0b8b72b2cba3c7f3510d385798/?
erid=CQH36pWzJqLuYYVAgpcTmGhSRPyyd5VGpNkQSD9shkKNGE",
    "playerId": "video_frame"
  }
}
```

Промо ролик Premier

Получаемые события от плеера через postMessage

player:promoStart

Описание

Старт проигрывания промо ролика. При получении события необходимо показать кнопки "Буду смотреть"/"Пропустить".

```
{
  "type": "player:promoStart",
  "instanceId": "123",
  "sender": "player",
  "data": {
    "onboardingId": 1111,
    "playerId": "video_frame"
  }
}
```

player:promoEnded

Описание

Событие отправляется плеером, после прекращения показа промо ролика. Примеры прекращения показа: ролик доиграл до конца, получили событие **player:promoInterrupt**, произошла ошибка во время воспроизведения ролика. При получении события необходимо скрывать кнопки "Буду смотреть"/"Пропустить".

```
{
  "type": "player:promoEnded",
  "instanceId": "123",
  "sender": "player",
  "data": {
    "onboardingId": 1111,
    "playerId": "video_frame"
  }
}
```

player:promoPlay

Промо-ролик запущен на воспроизведение (автоматически или пользователем).

```
{
      "type": "player:promoPlay",
      "instanceId": "playground-premier-1",
      "sender": "player",
      "data": {
        "onboardingId": 646627,
        "playerId": "video_frame"
      }
    }
```

player:promoPause

Промо-ролик поставлен на паузу пользователем.

```
{
      "type": "player:promoPause",
      "instanceId": "playground-premier-1",
      "sender": "player",
      "data": {
        "onboardingId": 646627,
        "playerId": "video_frame"
      }
    }
```

Отправляемые события в плеер через postMessage

player:promoInterrupt

Описание

Событие должно отправляться в плеер, когда необходимо скрыть текущий промо ролик. Пользователь нажал одну из кнопок "Буду смотреть"/" Пропустить".

Пример отправки события

```
window.postMessage(JSON.stringify({type: 'player:promoInterrupt', instanceId: 'TBD', sender: 'your_sender',
data: {}}), '*');
```

Пропуск рекламы

Получаемые события от плеера через postMessage

player:adSkippableStateChange

Описание

События отсылается, если мы получили от adsdk событие AdSkippableStateChange со значением skippableState: true, что означает, что реклама может быть пропущена.

Данные

```
 {
    "type": "player:adSkippableStateChange",
    "data": {
            "aformat": "preroll2", //   *preroll*, *midroll*, *postroll*
        "did": "6b6f8e301a48b6e61832140bd625fc47", // fingerprint id
        "videoId": "5551dc706488f4ab74c509b01dfc1b2e", //
            "e": "a_request",
            "axurl": "https://yandex.ru/ads/adfox/277740/getCode?p1=crunr&p2=gdol&pke=1&eid2=8:
5551dc706488f4ab74c509b01dfc1b2e&pk=2-
21824&puid20=5551dc706488f4ab74c509b01dfc1b2e&puid21=8&puid22=21824&puid23=4&puid24=2&puid25=2&puid26=7&dl=%%
extUrl%%",
            "playerId": "video_frame",
                "currentTime": 858,63,//
                "duration": 5984,23,//
        }
}
```

Отправляемые события в плеер через postMessage

player:skipAd

Описание

Событие должно отправляться в плеер, когда необходимо скрыть рекламу. Скрытие произойдет, если предварительно было получено AdSkippableStateChange от adsdk.

Пример отправки события

```
window.postMessage(JSON.stringify({type: 'player:skipAd', sender: 'other', data: {}}), '*');
```

События с данными от бекенда

Получаемые события от плеера через postMessage

player:activityConfig

Описание

События отсылается, когда мы получаем от бекенда ответ на запрос /api/v1/activity.

Данные

```
{
   "type": "player:activityConfig",
   "data": {
       "activities": [
           {
               "annotation_id": "7a19c4de-5c5a-4f30-af56-bffc38946bc9",
               "annotation_type": "SKIP_INTRO",
               "activity_id": "14",
               "activity_type": "BUTTON",
               "params": {
                   "on_click": {
                       "strategy": "HIDE"
                   },
                   "show": {
                       "strategy": "AT_TIME",
                       "end_position_ms": 50000,
                       "start_position_ms": 1000
                   },
                   "text": " "
               }
           },
           {
               "annotation_id": "7a19c4de-5c5a-4f30-af56-bffc38946bc9",
               "annotation_type": "SKIP_INTRO",
               "activity_id": "15",
               "activity_type": "BUTTON",
               "params": {
                   "on_click": {
                       "strategy": "SEEK_EXACTLY",
                       "target_position_ms": 50000
                   },
                   "show": {
                       "strategy": "AT_TIME",
                       "end_position_ms": 50000,
                       "start_position_ms": 1000
                   },
                   "text": ""
               }
           },
               ], //  activity
           "playerId": "video_frame",
       }
}
```