**Продукты OAG и способы получения данных.**

# Обзор API-интерфейсов OAG

OAG предоставляет набор API-интерфейсов, позволяющих организациям получать доступ к актуальной и стандартизированной авиационной информации. Все интерфейсы реализованы по REST-протоколу и возвращают данные в формате JSON. Ниже представлен перечень доступных API с кратким описанием их назначения и возможностей.

### Flight Info (v2) - Информация о рейсе

Ссылка на раздел на портале разработчика OAG

Интерфейс предназначен для получения информации о расписаниях и статусах рейсов. Поддерживается возврат дополнительных параметров, таких как прогнозируемое и фактическое количество пассажирских мест. Данные предоставляются на уровне конкретного рейса. API позволяет выполнять запросы как к отдельным рейсам, так и по маршрутам, с возможностью фильтрации по различным критериям (аэропорты, перевозчики, время и т.д.).

### Flight Info Alerts (v1) - Оповещения о рейсах

Ссылка на раздел на портале разработчика OAG

Интерфейс предоставляет доступ к событиям об изменениях расписаний и статусов рейсов в режиме, приближенном к реальному времени. Это позволяет отказаться от регулярных запросов к API и получать обновления через систему событий, настроенную на стороне клиента. Поддерживаются также дополнительные данные, включая информацию о количестве мест. Уведомления могут использоваться для автоматического оповещения агентов и пассажиров.

### Schedules – Расписания

Ссылка на раздел на портале разработчика OAG

API обеспечивает почти прямой доступ к полному спектру данных о расписаниях рейсов, хранящихся в системе OAG. Позволяет отображать актуальную доступность рейсов на витринах или использовать данные для построения маршрутов и анализа. Данные обновляются практически в реальном времени.

### Flight Info Connections (v1) - Информация о рейсах Связи

Ссылка на раздел на портале разработчика OAG

Предоставляет информацию о стыковочных маршрутах для пассажирских и грузовых перевозок на глобальном уровне. Обновление данных происходит еженедельно. Интерфейс учитывает минимальное время пересадки (MCT), надёжность соединений

и прочие отраслевые требования. Данные могут быть использованы для построения логистических цепочек и расчёта допустимых пересадок.

### Carriers (v1) – Перевозчики

Ссылка на раздел на портале разработчика OAG

Позволяет получить доступ к справочнику авиаперевозчиков с использованием кодов IATA и ICAO. API включает дополнительную информацию о перевозчике: наименование, страну регистрации, альянс и другие атрибуты. Является частью основного справочника для систем, работающих с глобальной авиационной информацией.

### Equipment (v1) – Оборудование

Ссылка на раздел на портале разработчика OAG

Содержит справочные данные об авиационном оборудовании и воздушных судах с кодами IATA и ICAO. Помимо кодов, API предоставляет дополнительную информацию, такую как конфигурации, модель и характеристики оборудования, используемого в рейсах.

### Locations (v1) – Местоположения

Ссылка на раздел на портале разработчика OAG

Позволяет получить доступ к международным кодам аэропортов IATA, ICAO и FAA. API возвращает наименование аэропорта, страну, регион, координаты, временную зону и другие описательные параметры. Используется для интеграции справочной информации в навигационные и маршрутные системы.

### Flight Info (v1.0) - Информация о рейсе

Ссылка на раздел на портале разработчика OAG

Первая версия интерфейса для получения расписаний и статусов. Считается устаревшей и выведена из эксплуатации 31 декабря 2024 года.

# Первые шаги

Для старта потребуется зарегистрироваться на портале разработчика OAG https://developers.oag.com/

Далее необходимо связаться со службой поддержки для активации тестового доступа или заключения договора с выбранным тарифом.

**Представитель OAG**
Имя: Megan Montgomery
E-mail: Megan.Montgomery@oag.com
Тел. +44 (0)7793 577 590
LinkedIn:смотреть профиль

**Инженер по решениям OAG**
Имя: Jade Bell
E-mail: Jade.Bell@oag.com

**Ключи с тестовым доступом до 8 апреля 2025г**

OAG_PRIMARY_KEY = b3a910c603c249cd9ed64517166fc3e1

OAG_SECONDARY_KEY = e0ede0cbc49a4f4b9df2eef39c726907

**Пример рабочего маршрута на Node.js для проверки статуса рейса (Flight Info-v2):**

```
app.get('/api/oag', async (req, res) => {
  console.log('[OAG] Запрос получен:', req.query);

  const { DepartureDateTime, CarrierCode, FlightNumber } = req.query;

  if (!DepartureDateTime || !CarrierCode || !FlightNumber) {
    console.warn('[OAG] Отсутствуют обязательные параметры');
    return res.status(400).json({ error: 'Missing required query parameters' });
  }

  const baseUrl = 'https://api.oag.com/flight-instances/';
  const queryParams = new URLSearchParams({
    DepartureDateTime,
    CarrierCode,
    FlightNumber,
    CodeType: 'IATA,ICAO',
    Content: 'status',
    version: 'v2'
  });

  const url = `${baseUrl}?${queryParams}`;
  console.log('[OAG] Полный URL:', url);

  // Диагностика Axios - логирование всех запросов
  axios.interceptors.request.use((config) => {
    console.log('[Axios] Отправляем запрос:', config.method.toUpperCase(), config.url);
    console.log('[Axios] Заголовки запроса:', config.headers);
    return config;
  });

  const tryRequest = async (keyLabel, apiKey) => {
    console.log(`[OAG] Используем ключ: ${keyLabel}`);
    return axios.get(url, {
      headers: {
```

```
        'Subscription-Key': apiKey,
        'Cache-Control': 'no-cache'
      }
    });
  };

  try {
    let response;
    try {
      response = await tryRequest('PRIMARY', process.env.OAG_PRIMARY_KEY);
    } catch (err) {
      console.warn('[OAG] PRIMARY ключ не сработал:', err.response?.data || err.message);

      if (process.env.OAG_SECONDARY_KEY) {
        console.warn('[OAG] Пробуем SECONDARY ключ...');
        response = await tryRequest('SECONDARY', process.env.OAG_SECONDARY_KEY);
      } else {
        throw err;
      }
    }

    console.log(`[OAG] Ответ получен, статус: ${response.status}`);
    res.json(response.data);
  } catch (error) {
    console.error('[OAG] Ошибка запроса:', error.response?.data || error.message);
    res.status(500).json({ error: 'Failed to fetch OAG data' });
  }
});
```

## Пример отправки данных с клиента

## Поля ввода

```
    <div id="inputFields">
    <label>
      Carrier Code:
      <input type="text" name="carrierCode" placeholder="e.g. FZ" required>
    </label>
    <label>
      Flight Number:
      <input type="text" name="flightNumber" placeholder="e.g. 241" required>
    </label>
    <label>
      Day:
      <input type="number" name="day" placeholder="e.g. 31" required>
    </label>
    <label>
      Month:
      <input type="number" name="month" placeholder="e.g. 3" required>
    </label>
    <label>
      Year:
      <input type="number" name="year" placeholder="e.g. 2025" required>
    </label>
  </div>
    <button onclick="fetchOAG()">Check Flight :: OAG</button>
```

## Область вывода ответа

```
  <h3>Response OAG</h3>
  <div id="oagTimeTaken" class="response-time"></div>
  <div id="resultOAG" class="styled-table-container">Data here</div>
```

## Клиент JS

```
<script>
    const timeDiv = document.getElementById('timeTaken');

    // Устанавливаем текущую дату по умолчанию
    window.addEventListener('DOMContentLoaded', () => {
    const now = new Date();
    document.querySelector('[name="day"]').value = now.getDate();
    document.querySelector('[name="month"]').value = now.getMonth() + 1;
    document.querySelector('[name="year"]').value = now.getFullYear();
    });

    function getFormData() {
    const inputs = document.querySelectorAll('#inputFields input');
    return Object.fromEntries([...inputs].map(i => [i.name, i.value]));
  }

 async function fetchOAG() {
  const formData = getFormData();

  if (!formData.carrierCode || !formData.flightNumber || !formData.year || !formData.month ||
!formData.day) {
    alert('Пожалуйста, заполните все поля.');
    return;
  }

  const query = {
    CarrierCode: formData.carrierCode,
    FlightNumber: formData.flightNumber,
    DepartureDateTime: `${formData.year}-${String(formData.month).padStart(2, '0')}-
${String(formData.day).padStart(2, '0')}`
  };

  const params = new URLSearchParams(query).toString();
  const url = `https://ai.kupi.com:3001/api/oag?${params}`;

  const start = performance.now();

  try {
    const res = await fetch(url);
    const json = await res.json();

    const duration = performance.now() - start;
    document.getElementById('oagTimeTaken').textContent = `⏱ Ответ за ${Math.round(duration)}
мс`;

    if (!res.ok) {
      document.getElementById('resultOAG').textContent = `Ошибка ${res.status}: ${json.error ||
'Неизвестная ошибка'}`;
      return;
    }

    renderTable(json, 'resultOAG');
  } catch (err) {
    document.getElementById('resultOAG').textContent = 'Ошибка запроса: ' + err;
  }
}

  function pad(n) {
    return n.toString().padStart(2, '0');
  }

  function renderTable(data, targetId = 'result') {
  const container = document.getElementById(targetId);
  container.innerHTML = ''; // очищаем старое

  if (!data || typeof data !== 'object') {
    container.textContent = 'Невозможно отобразить данные';
    return;
  }
```

```
  const createRows = (obj, prefix = '') => {
    return Object.entries(obj).flatMap(([key, value]) => {
      const fullKey = prefix ? `${prefix}.${key}` : key;
      if (typeof value === 'object' && value !== null && !Array.isArray(value)) {
        return createRows(value, fullKey);
      } else if (Array.isArray(value)) {
        return value.length > 0 && typeof value[0] === 'object'
          ? value.flatMap((item, idx) => createRows(item, `${fullKey}[${idx}]`))
          : [[fullKey, JSON.stringify(value)]];
      } else {
        return [[fullKey, value]];
      }
    });
  };

  const rows = createRows(data);
  const table = document.createElement('table');
  table.innerHTML = `<thead><tr><th>Поле</th><th>Значение</th></tr></thead>`;

  const tbody = document.createElement('tbody');
  for (const [key, val] of rows) {
    const row = document.createElement('tr');
    row.innerHTML = `<td>${key}</td><td>${val}</td>`;
    tbody.appendChild(row);
  }

  table.appendChild(tbody);
  container.appendChild(table);
}

  function flatten(obj, prefix = '') {
    return Object.entries(obj).reduce((acc, [k, v]) => {
      const key = prefix ? `${prefix}.${k}` : k;
      if (typeof v === 'object' && v !== null && !Array.isArray(v)) {
        Object.assign(acc, flatten(v, key));
      } else {
        acc[key] = Array.isArray(v) ? JSON.stringify(v) : v;
      }
      return acc;
    }, {});
  }
  </script>
```

Вышеуказанный пример позволит раскидать полученный ответ по строкам таблицы.

По аналогии работает API из других разделов. Для доступа к каждому из разделов используется индивидуальная пара ключей.