FChat Webhook Integration Documentation

Table of Contents

Quick Start

Base URL

```
https://alerts.soc.fpt.net/webhooks
```

Available Endpoints

Endpoint	URL	Purpose
Group Messaging	`/{app_name}/{token}`	Send to pre-configured FChat groups
User Messaging	`/{token}/fchat`	Send to individual FChat users

What Can Each Endpoint Do?

Feature	Group Messaging	User Messaging
Send to groups	✅ (pre-configured only)	❌

Feature	Group Messaging	User Messaging
Send to individual users	❌	✅
Interactive buttons	❌	✅
Auto-create users	❌	✅

Endpoint 1: Group Messaging

URL: `POST /{app_name}/{token}`

Overview

Send messages to FChat groups that are pre-configured in your application settings. The group must be assigned and active.

Authentication

- `app_name` - Your application name (URL path)
- `token` - Your application token (URL path)

Request Body

```json
{
  "text": "Your message here"  // Max 2000 characters
}
```

Quick Example

```
curl -X POST https://alerts.soc.fpt.net/webhooks/MY_APP/MY_TOKEN \
  -H "Content-Type: application/json" \
  -d '{"text": "System notification: All services running"}'
```

Success Response

```json
{
  "ok": true
}
```

Common Errors

Code	Reason	Solution
404	Invalid app_name or token	Verify credentials
404	No group assigned	Assign a group to your app
400	Group not active	Activate the group
400	Empty message	Include text in request
400	Message too long	Keep under 2000 characters

Prerequisites

- Application must have a group assigned (`application.group_id`)
- Group must be active
- Group must have a valid `beatchat_id`

Limitations

- ❌ Cannot send to arbitrary groups
- ❌ Cannot specify `beatchat_id` in request
- ✅ Only works with pre-configured groups

Endpoint 2: User Messaging

URL: `POST /{token}/fchat`

Overview

Send messages to individual FChat users via email. Users are automatically created if they don't exist. Supports interactive buttons.

Authentication

- `token` - Your application token (URL path)

Request Body

Simple Message

```json
{
  "email": "user@example.com",
  "text": "Your message here"  // Max 2000 characters
}
```

With Interactive Buttons

```json
{
  "email": "user@example.com",
  "text": "Please choose an option:",
  "reply_markup": {
    "inlineKeyboard": [
      [
        {
          "text": "✅ Approve",
          "callbackData": {
            "action": "approve",
            "data": "approve_request"
          }
        },
        {
          "text": "❌ Reject",
          "callbackData": {
            "action": "reject",
            "data": "reject_request"
```

```
          }
        }
      ]
    ]
  }
}
```

Legacy Button Format (Backward Compatible)

```json
{
  "email": "user@example.com",
  "text": "Please choose:",
  "buttons": [
    {"title": "Option A", "payload": "option_a"},
    {"title": "Option B", "payload": "option_b"}
  ]
}
```

Quick Examples

Simple Message

```
curl -X POST https://alerts.soc.fpt.net/webhooks/MY_TOKEN/fchat \
  -H "Content-Type: application/json" \
  -d '{"email": "user@example.com", "text": "Hello!"}'
```

With Buttons

```
curl -X POST https://alerts.soc.fpt.net/webhooks/MY_TOKEN/fchat \
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "text": "Approve this request?",
    "reply_markup": {
      "inlineKeyboard": [[
        {"text": "Yes", "callbackData": {"action": "confirm", "data": "yes"}},
        {"text": "No", "callbackData": {"action": "confirm", "data": "no"}}
      ]]
```

```
    }
  }'
```

Success Response

```json
{
  "ok": true,
  "error_code": 200,
  "description": "Message sent to queue",
  "context": {
    "id": "context_id_12345",
    "parameters": {}
  }
}
```

Common Errors

Code	Reason	Solution
404	Invalid token	Verify your token
404	No permission to send	Check app permissions
400	Empty email	Include email in request
400	Empty message	Include text in request
400	Message too long	Keep under 2000 characters
400	Too many buttons	Max 30 buttons per message
400	Invalid reply_markup	Check format structure

Limitations

- ❌ Cannot send to groups
- ❌ No `beatchat_id` support
- ✅ Only works with individual users via email

Incoming Webhooks

Overview

When FChat users interact with your messages (send text or click buttons), Iris forwards the event to your configured webhook URL.

Webhook Payload Structure

```json
{
  "message": {
    "text": "User message content",
    "user": {
      "email": "user@example.com",
      "name": "User Name"
    },
    "chat": {
      "id": "chat_id",
      "type": "private"
    },
    "callback_query": {
      "data": "{\"action\":\"approve\",\"data\":\"approve_request\"}",
      "message": {
        "message_id": "message_id"
      }
    }
  }
}
```

Your Response

Your webhook should respond with HTTP 200:

```json
{
  "ok": true
}
```

Callback Data Structure

When a button is clicked, `callback_query.data` contains the JSON string you defined in `callbackData`:

```json
{
  "action": "approve",
  "data": "approve_request"
}
```

Parse this to handle button interactions in your application.

Reference

Message Flow Diagrams

▶ **Group Messaging Flow**
▶ **User Messaging Flow**
▶ **Incoming Webhook Flow**

Key Features

Message Queue

- All messages are queued for processing
- API response indicates successful queuing, not immediate delivery

User Auto-Creation

- Users are automatically created if they don't exist
- Only applies to User Messaging endpoint

Context Management

- Each message generates a context ID
- Use for message tracking, replies, and conversation state
- Default expiration: 5 minutes (configurable)

Rate Limits

- Message length: 2000 characters (both endpoints)
- Button limit: 30 buttons per message
- Context expiration: 5 minutes (default)

Error Handling

- Consistent error format across all endpoints
- `ok`: Boolean success/failure indicator
- `error_code`: HTTP status code
- `description`: Human-readable error message

Logging

- All requests are logged for monitoring and debugging

Full Field Reference

Group Messaging Request

Field	Type	Required	Description	Constraints
`text`	string	Yes*	Message text content	Max 2000 characters
`message`	string	Yes*	Alternative message field	Max 2000 characters

*Either `text` or `message` is required

User Messaging Request

Field	Type	Required	Description	Constraints
`email`	string	Yes	Recipient's email	Valid email format
`text`	string	Yes*	Message text content	Max 2000 characters
`message`	string	Yes*	Alternative message field	Max 2000 characters

Field	Type	Required	Description	Constraints
`reply_markup`	object	No	Inline keyboard markup	See button structure
`buttons`	array	No	Legacy button format	Max 30 buttons

*Either `text` or `message` is required

Complete Error Reference

Group Messaging Errors

Status	Description
400	Bad Request: message text is empty
400	Bad Request: message is too long
400	Bad Request: beatchat group id not found
400	Bad Request: group is not active
404	Channel or token not found
404	Application does not have a group assigned
404	Group not found

User Messaging Errors

Status	Description
400	Bad Request: invalid request
400	Bad Request: email is empty
400	Bad Request: message text is empty
400	Bad Request: message is too long
400	Bad Request: reply_markup must be a dictionary

Status	Description
400	Bad Request: inlineKeyboard must be a list of list of dicts
400	Limit 30 buttons
404	channel or token not found
404	this app have not permission to send fchat message
404	User {email} not found!

Need Help? Contact your system administrator or check the Iris admin panel for your application credentials.