CX-Catalyst - API Reference

Complete technical reference for all webhook endpoints and integration points.

Table of Contents

Overview

Base URL

`Production: https://your-n8n-instance.com/webhook`

Content Type

All requests must use:

`Content-Type: application/json`

Versioning

Current version: **v1** (implicit in endpoints)

Authentication

Webhook Security

Webhooks can be secured using:

1. **Path Tokens** - Include secret in webhook path
2. **Header Authentication** - Custom header validation
3. **IP Allowlisting** - Restrict source IPs

Path Token Example

`https://your-n8n.com/webhook/support/intake/abc123secret`

Header Authentication Example

```
curl -X POST https://your-n8n.com/webhook/support/intake \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your-api-key" \
  -d '{"customer_id": "..."}'
```

Webhook Endpoints

1. Support Intake

Create a new support case.

Endpoint: `POST /webhook/support/intake`

Workflow: 1 - Smart Intake & Triage

Request

```
{
  "customer_id": "string (UUID, optional)",
  "customer_email": "string (required)",
  "customer_name": "string (required)",
  "description": "string (required, max 10000 chars)",
  "severity": "string (optional: critical|high|medium|low)",
  "product": "string (optional)",
  "product_version": "string (optional)",
  "environment": "string (optional: production|staging|development)",
  "channel": "string (optional: webhook|email|chat|portal)",
  "metadata": {
    "user_agent": "string (optional)",
    "ip_address": "string (optional)",
    "session_id": "string (optional)",
    "custom_field": "any (optional)"
  }
}
```

Response (Success: 200)

```
{
  "success": true,
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "request_id": "REQ-20260115-abc123",
  "status": "triaged",
  "classification": {
    "category": "configuration",
    "subcategory": "ssl-certificates",
    "priority": "high",
    "confidence": 0.87,
    "estimated_complexity": "simple",
    "resolution_path": "self-service"
  },
  "message": "Your support request has been received and is being processed.",
  "estimated_response_time": "Within 15 minutes",
  "tracking_url": "https://portal.company.com/cases/550e8400..."
}
```

Response (Error: 400)

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Missing required field: customer_email",
    "details": {
      "field": "customer_email",
      "requirement": "Valid email address required"
    }
  }
}
```

---

2. Self-Service Trigger

Manually trigger self-service resolution for a case.

Endpoint: `POST /webhook/support/self-service`

Workflow: 2 - Self-Service Resolution Engine

Request

```
{
  "case_id": "string (UUID, required)",
  "customer_id": "string (UUID, required)",
  "force": "boolean (optional, default: false)"
}
```

Response (Success: 200)

```
{
  "success": true,
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "in_progress",
  "resolution": {
    "solution_provided": true,
    "solution_type": "automated",
    "estimated_time": 5,
    "steps_count": 4
  },
  "message": "Self-service resolution initiated"
}
```

---

3. Solution Approval

Approve automated solution execution.

Endpoint: `GET /webhook/support/approve/:caseId`

Workflow: 2 - Self-Service Resolution Engine

URL Parameters

Parameter	Type	Description
caseId	UUID	Case identifier

Query Parameters

Parameter	Type	Description
action	string	`approve` or `reject`
token	string	Security token (optional)

Example

```
GET /webhook/support/approve/550e8400-e29b-41d4-a716-446655440000?action=approve
```

Response (Success: 200)

```
{
  "success": true,
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "action": "approve",
  "status": "executing",
  "message": "Automated fix is being applied. You will receive confirmation
shortly."
}
```

---

4. Customer Feedback

Submit satisfaction feedback for resolved case.

Endpoint: `GET /webhook/support/feedback/:caseId`

Workflow: 2 - Self-Service Resolution Engine

URL Parameters

Parameter	Type	Description
caseId	UUID	Case identifier

Query Parameters

Parameter	Type	Description
score	integer	1-5 satisfaction rating
comment	string	Optional feedback text (URL encoded)

Example

```
GET /webhook/support/feedback/550e8400...?score=5&comment=Great%20service
```

Response (Success: 200)

```
{
  "success": true,
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "feedback_received": true,
  "message": "Thank you for your feedback!"
}
```

---

5. Collaborative Support Trigger

Initiate collaborative (human-in-loop) resolution.

Endpoint: `POST /webhook/support/collaborative`

Workflow: 4 - Collaborative Support Hub

Request

```
{
  "case_id": "string (UUID, required)",
  "customer_id": "string (UUID, required)",
  "urgency": "string (optional: urgent|normal)",
  "specialist_request": "string (optional)",
  "additional_context": "string (optional)"
}
```

Response (Success: 200)

```
{
  "success": true,
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "review_id": "rev-abc123",
  "status": "pending_review",
  "assigned_channel": "#support-review",
  "estimated_review_time": "Within 2 hours",
  "message": "Your case has been escalated for specialist review."
}
```

---

6. Human Review Action

Submit human review decision.

Endpoint: `GET /webhook/support/review/:reviewId/:action`

Workflow: 4 - Collaborative Support Hub

URL Parameters

Parameter	Type	Description
reviewId	string	Review queue identifier
action	string	`approve`, `edit`, or `reject`

Query Parameters

Parameter	Type	Description
reviewer	string	Reviewer email/ID
comments	string	Review comments (URL encoded)

Example - Approve

```
GET /webhook/support/review/rev-abc123/approve?reviewer=jane@company.com
```

Example - Edit

For edits, use POST with body:

```
POST /webhook/support/review/rev-abc123/edit

{
  "reviewer": "jane@company.com",
  "edited_solution": "Updated solution text...",
  "edit_reason": "Added clarification on step 3"
}
```

Response (Success: 200)

```
{
  "success": true,
  "review_id": "rev-abc123",
  "case_id": "550e8400-e29b-41d4-a716-446655440000",
  "action": "approve",
  "status": "delivering",
  "message": "Solution approved and being delivered to customer."
}
```

Outgoing Webhooks

CX-Catalyst can send webhook notifications to external systems when key events occur. Configure outgoing webhook URLs in n8n workflow settings or environment variables.

Event Types

Event	Trigger	Payload Key Fields
`case.created`	New case submitted and triaged	case_id, classification, priority
`case.resolved`	Case resolved (self-service or human)	case_id, resolution_type, resolution_time_minutes
`case.escalated`	Case escalated to human review	case_id, escalation_reason, assigned_channel
`review.completed`	Human review action taken	review_id, case_id, action (approve/edit/reject)
`feedback.received`	Customer feedback submitted	case_id, score, comment
`alert.triggered`	Proactive alert detected	alert_id, severity, description

Outgoing Webhook Payload Format

All outgoing webhooks use the following envelope:

```
{
  "event": "case.resolved",
  "timestamp": "2026-01-15T10:30:00Z",
  "workflow": "workflow-2-self-service",
  "data": {
    "case_id": "550e8400-e29b-41d4-a716-446655440000",
    "customer_id": "customer-uuid",
    "resolution_type": "self-service-automated",
    "resolution_time_minutes": 2,
    "satisfaction_score": null,
    "metadata": {}
  }
}
```

Configuring Outgoing Webhooks

In n8n, outgoing webhooks are configured via HTTP Request nodes at the end of workflow branches. To add a new outgoing webhook destination:

1. Open the relevant workflow in n8n
2. Add an **HTTP Request** node after the event trigger point
3. Set **Method** to `POST`
4. Set **URL** to your receiving endpoint
5. Configure **Headers** (e.g., `X-Webhook-Secret` for signature verification)
6. Map the payload fields from upstream nodes

Webhook Signature Verification

Outgoing webhooks include an `X-Webhook-Signature` header containing an HMAC-SHA256 signature of the payload body. Verify the signature on your receiving end to ensure authenticity:

```
const crypto = require('crypto');

function verifySignature(payload, signature, secret) {
    const expected = crypto
        .createHmac('sha256', secret)
        .update(JSON.stringify(payload))
        .digest('hex');
    return crypto.timingSafeEqual(
        Buffer.from(signature),
        Buffer.from(expected)
    );
}
```

Retry Policy

Failed outgoing webhook deliveries are retried up to 3 times with exponential backoff (1s, 5s, 30s). After all retries are exhausted, the failure is logged in the `workflow_executions` table.

Request/Response Formats

Standard Success Response

```
{
  "success": true,
  "data": { ... },
  "message": "Human-readable status message",
  "timestamp": "2026-01-15T10:30:00Z"
}
```

Standard Error Response

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "details": { ... }
  },
  "timestamp": "2026-01-15T10:30:00Z"
}
```

Data Types

Type	Format	Example
UUID	RFC 4122	`550e8400-e29b-41d4-a716-446655440000`
Timestamp	ISO 8601	`2026-01-15T10:30:00Z`
Email	RFC 5322	`user@example.com`
Priority	Enum	`critical`, `high`, `medium`, `low`
Status	Enum	`new`, `triaged`, `in_progress`, `resolved`, `closed`

Error Handling

Error Codes

Code	HTTP Status	Description
`VALIDATION_ERROR`	400	Invalid request data
`MISSING_FIELD`	400	Required field not provided
`INVALID_FORMAT`	400	Field format incorrect
`NOT_FOUND`	404	Case or resource not found
`ALREADY_PROCESSED`	409	Action already taken
`RATE_LIMITED`	429	Too many requests
`INTERNAL_ERROR`	500	System error
`SERVICE_UNAVAILABLE`	503	Dependency unavailable

Error Response Examples

Validation Error

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Request validation failed",
    "details": {
      "fields": [
        {
          "field": "customer_email",
          "error": "Invalid email format"
        },
        {
          "field": "description",
```

```
          "error": "Description must be at least 10 characters"
        }
      ]
    }
  }
}
```

Not Found

```
{
  "success": false,
  "error": {
    "code": "NOT_FOUND",
    "message": "Case not found",
    "details": {
      "case_id": "550e8400-e29b-41d4-a716-446655440000",
      "suggestion": "Verify the case ID is correct"
    }
  }
}
```

Rate Limited

```
{
  "success": false,
  "error": {
    "code": "RATE_LIMITED",
    "message": "Too many requests",
    "details": {
      "limit": 100,
      "window": "1 minute",
      "retry_after": 45
    }
  }
}
```

Rate Limits

Default Limits

Endpoint	Limit	Window
`/intake`	100 requests	1 minute
`/self-service`	50 requests	1 minute
`/approve`	200 requests	1 minute
`/feedback`	200 requests	1 minute
`/collaborative`	50 requests	1 minute
`/review`	100 requests	1 minute

Rate Limit Headers

Responses include:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1704803460
```

Integration Examples

cURL

Create Case

```
curl -X POST https://your-n8n.com/webhook/support/intake \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your-api-key" \
  -d '{
    "customer_email": "john.doe@example.com",
    "customer_name": "John Doe",
    "description": "Cannot login to my account. Getting error code ERR_AUTH_001
when trying to authenticate.",
    "severity": "high",
    "product": "web-portal",
    "environment": "production"
  }'
```

Python

```
import requests
import json

BASE_URL = "https://your-n8n.com/webhook"
API_KEY = "your-api-key"

def create_support_case(email, name, description, severity="medium"):
    """Create a new support case."""
    response = requests.post(
        f"{BASE_URL}/support/intake",
        headers={
            "Content-Type": "application/json",
            "X-API-Key": API_KEY
        },
        json={
            "customer_email": email,
            "customer_name": name,
            "description": description,
            "severity": severity
        }
    )
    return response.json()
```

```python
def get_case_status(case_id):
    """Check status of a case."""
    response = requests.get(
        f"{BASE_URL}/support/status/{case_id}",
        headers={"X-API-Key": API_KEY}
    )
    return response.json()

def submit_feedback(case_id, score, comment=None):
    """Submit feedback for resolved case."""
    params = {"score": score}
    if comment:
        params["comment"] = comment

    response = requests.get(
        f"{BASE_URL}/support/feedback/{case_id}",
        params=params
    )
    return response.json()

# Usage
result = create_support_case(
    email="user@example.com",
    name="Test User",
    description="Unable to reset password",
    severity="medium"
)
print(f"Case created: {result['case_id']}")
```

JavaScript/Node.js

```javascript
const axios = require('axios');

const BASE_URL = 'https://your-n8n.com/webhook';
const API_KEY = 'your-api-key';

async function createSupportCase(email, name, description, severity = 'medium') {
  try {
    const response = await axios.post(
      `${BASE_URL}/support/intake`,
      {
        customer_email: email,
        customer_name: name,
        description: description,
        severity: severity
      },
      {
        headers: {
          'Content-Type': 'application/json',
```

```
          'X-API-Key': API_KEY
        }
      }
    );
    return response.data;
  } catch (error) {
    if (error.response) {
      throw new Error(`API Error: ${error.response.data.error.message}`);
    }
    throw error;
  }
}

async function approveResolution(caseId) {
  const response = await axios.get(
    `${BASE_URL}/support/approve/${caseId}`,
    {
      params: { action: 'approve' },
      headers: { 'X-API-Key': API_KEY }
    }
  );
  return response.data;
}

// Usage
(async () => {
  const result = await createSupportCase(
    'user@example.com',
    'Test User',
    'Page loads slowly on dashboard'
  );
  console.log('Case ID:', result.case_id);
})();
```

PHP

```
<?php

class SupportAPI {
    private $baseUrl;
    private $apiKey;

    public function __construct($baseUrl, $apiKey) {
        $this->baseUrl = $baseUrl;
        $this->apiKey = $apiKey;
    }

    public function createCase($email, $name, $description, $severity = 'medium')
{
        $data = [
```

```php
            'customer_email' => $email,
            'customer_name' => $name,
            'description' => $description,
            'severity' => $severity
        ];

        $ch = curl_init($this->baseUrl . '/support/intake');
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));
        curl_setopt($ch, CURLOPT_HTTPHEADER, [
            'Content-Type: application/json',
            'X-API-Key: ' . $this->apiKey
        ]);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        $response = curl_exec($ch);
        curl_close($ch);

        return json_decode($response, true);
    }

    public function submitFeedback($caseId, $score, $comment = null) {
        $url = $this->baseUrl . '/support/feedback/' . $caseId;
        $url .= '?score=' . $score;
        if ($comment) {
            $url .= '&comment=' . urlencode($comment);
        }

        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_HTTPHEADER, [
            'X-API-Key: ' . $this->apiKey
        ]);

        $response = curl_exec($ch);
        curl_close($ch);

        return json_decode($response, true);
    }
}

// Usage
$api = new SupportAPI('https://your-n8n.com/webhook', 'your-api-key');
$result = $api->createCase(
    'user@example.com',
    'Test User',
    'Login issues with SSO'
);
echo "Case ID: " . $result['case_id'];
```

Webhook Payload Signature (Optional)

For added security, validate webhook signatures:

```
const crypto = require('crypto');

function verifyWebhookSignature(payload, signature, secret) {
    const expectedSignature = crypto
        .createHmac('sha256', secret)
        .update(JSON.stringify(payload))
        .digest('hex');

    return crypto.timingSafeEqual(
        Buffer.from(signature),
        Buffer.from(expectedSignature)
    );
}

// In your webhook handler
app.post('/webhook/callback', (req, res) => {
    const signature = req.headers['x-webhook-signature'];

    if (!verifyWebhookSignature(req.body, signature, WEBHOOK_SECRET)) {
        return res.status(401).json({ error: 'Invalid signature' });
    }

    // Process webhook...
});
```

Appendix: Complete Schema Reference

Case Object

```
{
  "case_id": "UUID",
  "customer_id": "UUID",
  "channel": "string (webhook|email|chat|portal)",
  "description": "string",
  "category": "string (configuration|usage|setup|defect|enhancement|other)",
  "subcategory": "string",
  "priority": "string (critical|high|medium|low)",
  "confidence_score": "number (0.00-1.00)",
  "status": "string (new|triaged|in_progress|pending_review|resolved|closed|
escalated)",
  "resolution_type": "string (self-service-automated|self-service-manual|
collaborative|escalated)",
  "escalated": "boolean",
  "created_at": "timestamp",
  "triaged_at": "timestamp",
  "resolved_at": "timestamp",
```

```
  "closed_at": "timestamp",
  "resolution_time_minutes": "integer",
  "satisfaction_score": "integer (1-5)",
  "metadata": "object"
}
```

Classification Object

```
{
  "category": "string",
  "subcategory": "string",
  "priority": "string",
  "confidence": "number (0.00-1.00)",
  "estimated_complexity": "string (simple|moderate|complex)",
  "suggested_resolution_path": "string (self-service|collaborative|escalate)",
  "reasoning": "string",
  "tags": ["string"],
  "similar_cases": ["UUID"]
}
```

Solution Object

```
{
  "solution_steps": ["string"],
  "diagnostic_commands": ["string"],
  "automated_fix_available": "boolean",
  "estimated_time": "integer (minutes)",
  "confidence": "number (0.00-1.00)",
  "references": [
    {
      "type": "string (kb_article|documentation|case)",
      "id": "string",
      "title": "string",
      "url": "string"
    }
  ],
  "risks": ["string"],
  "requires_approval": "boolean",
  "reasoning": "string"
}
```
