NCM Webhook Integration Documentation

Overview

NCM (Nepal Can Move) provides webhook functionality to automatically notify vendors about order status updates. Webhooks are HTTP POST requests sent to your configured URL when specific events occur in the NCM system.

Supported Events

Order Status Events

The following order status changes trigger webhook notifications:

Event	Status	Description
`pickup_completed`	Pickup Complete	Order has been picked up from the origin
`sent_for_delivery`	Sent for Delivery	Order has been sent out for delivery
`order_dispatched`	Dispatched	Order has been dispatched from origin branch
`order_arrived`	Arrived	Order has arrived at destination branch
`delivery_completed`	Delivered	Order has been successfully delivered

Webhook Configuration

Setting Up Webhook URL

1. Log in to your NCM vendor portal
2. Navigate to API section (`/accounts/vendor/api`)
3. Scroll to "Webhook/API Callback URLs" section
4. Enter your webhook URL in the "Order Delivery Webhook URL" field
5. Click "Test Webhook" to verify your endpoint
6. Click "Add/Update" to save your configuration

URL Requirements

- Must be a valid HTTP/HTTPS URL
- Should respond to POST requests
- Recommended to use HTTPS for security
- URL can include query parameters if needed

Example webhook URLs:

```plaintext
https://your-domain.com/webhooks/ncm
https://your-domain.com/api/ncm-webhook?token=your-secret
```

Webhook Payload Structure

Single Order Webhook

For individual order events (like delivery completion):

```json
{
    "order_id": "123456",
    "status": "Delivered",
    "timestamp": "2024-01-15T10:30:00Z",
    "event": "delivery_completed"
}
```

Bulk Order Webhook

For bulk status updates (multiple orders):

```json
{
    "order_ids": ["123456", "123457", "123458"],
    "status": "Dispatched",
    "timestamp": "2024-01-15T10:30:00Z",
    "event": "order_dispatched"
}
```

Payload Fields

Field	Type	Description
`order_id`	string	Unique order identifier (single order)
`order_ids`	array	Array of order identifiers (bulk orders)
`status`	string	Current order status
`timestamp`	string	ISO 8601 formatted timestamp
`event`	string	Event type identifier

Request Headers

All webhook requests include the following headers:

```plaintext
Content-Type: application/json
Content-Length: [payload-length]
User-Agent: NCM-Webhook/1.0
```

Implementation Guidelines

Response Handling

- Your endpoint should respond quickly (within 10 seconds)
- Return any 2xx status code to acknowledge receipt
- NCM does not require specific response content
- Failed webhook deliveries are silently retried

Security Considerations

1. **URL Security**: Use HTTPS to encrypt webhook data
2. **Authentication**: Consider using query parameters or headers for authentication
3. **Rate Limiting**: Be prepared to handle multiple webhook requests
4. **Idempotency**: Design your endpoint to handle duplicate webhooks safely

Error Handling

- NCM implements silent error handling
- Webhook failures do not affect order processing
- No retry mechanism is currently implemented
- Monitor your webhook endpoint for connectivity issues

Testing Your Webhook

Manual Testing

Use the "Test Webhook" button in the NCM vendor portal to send a test payload:

```json
{
    "event": "order.status.changed",
    "order_id": "TEST-123456",
    "status": "In Transit",
    "timestamp": "2024-01-15T10:30:00Z",
    "test": true
}
```

Sample Implementation

Python (Flask)

python

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/webhooks/ncm', methods=['POST'])
def ncm_webhook():
    try:
        data = request.get_json()

        # Verify webhook (optional)
        if data.get('test'):
            print("Test webhook received")
            return jsonify({'status': 'success'})

        # Process order status update
        order_id = data.get('order_id') or data.get('order_ids', [])
        status = data.get('status')
        event = data.get('event')
        timestamp = data.get('timestamp')

        # Your business logic here
        print(f"Order {order_id} status updated to: {status}")

        return jsonify({'status': 'received'})

    except Exception as e:
        print(f"Webhook error: {e}")
        return jsonify({'error': 'processing failed'}), 500

if __name__ == '__main__':
    app.run(port=5000)
```

Node.js (Express)

javascript

```javascript
const express = require('express');
const app = express();

app.use(express.json());

app.post('/webhooks/ncm', (req, res) => {
    try {
        const { order_id, order_ids, status, event, timestamp, test } = req.body;

        // Handle test webhooks
        if (test) {
            console.log('Test webhook received');
            return res.json({ status: 'success' });
        }

        // Process order update
        const orders = order_id || order_ids;
        console.log(`Order(s) ${orders} status updated to: ${status}`);

        // Your business logic here

        res.json({ status: 'received' });

    } catch (error) {
        console.error('Webhook error:', error);
        res.status(500).json({ error: 'processing failed' });
    }
});

app.listen(3000, () => {
    console.log('Webhook server running on port 3000');
});
```

PHP

```php
<?php
header('Content-Type: application/json');

try {
    $json = file_get_contents('php://input');
    $data = json_decode($json, true);

    if (!$data) {
        throw new Exception('Invalid JSON');
    }

    // Handle test webhooks
    if (isset($data['test']) && $data['test']) {
        echo json_encode(['status' => 'success']);
        exit;
    }

    // Process order update
    $order_id = $data['order_id'] ?? $data['order_ids'] ?? [];
    $status = $data['status'] ?? '';
    $event = $data['event'] ?? '';
    $timestamp = $data['timestamp'] ?? '';

    // Your business logic here
    error_log("Order(s) " . implode(',', (array)$order_id) . " status updated to: $

    echo json_encode(['status' => 'received']);

} catch (Exception $e) {
    error_log("Webhook error: " . $e->getMessage());
    http_response_code(500);
    echo json_encode(['error' => 'processing failed']);
}
?>
```

Troubleshooting

Common Issues

1. **No Webhook Received**
    - Check if webhook URL is correctly configured
    - Verify your endpoint is accessible from the internet

    - Ensure your server responds to POST requests

2. **Timeout Errors**
    - Optimize your endpoint response time
    - Implement asynchronous processing
    - Check server performance and network connectivity

3. **Invalid Payload**
    - Ensure your endpoint can handle JSON content
    - Check for proper request header handling
    - Validate payload structure in your code

Debugging Tips

- Log all incoming webhook requests
- Check server logs for errors
- Use webhook testing tools like webhook.site for initial testing
- Monitor your endpoint's accessibility and response times

Support

For integration assistance and technical support:

- **Email**: IT@nepalcanmove.com
- **Documentation**: Available in the vendor portal
- **Testing**: Use the built-in webhook test functionality

Version Information

- **Current Version**: 1.0
- **User-Agent**: NCM-Webhook/1.0
- **Status**: Beta (subject to changes)

Last updated: January 2024