# Notification System

Comprehensive guide to the Notification system implemented in the RAS Dashboard API, providing multi-channel delivery, template management, and advanced analytics for complete notification lifecycle management.

## 🎯 Overview

The Notification system provides: - **Multi-Channel Delivery** - Email, SMS, push, webhook, Slack, Teams, Discord, in-app - **Template System** - Variable substitution with conditional logic - **Priority Management** - 4-level priority system with urgency handling - **Real-time Analytics** - Comprehensive notification statistics and reporting - **User Preferences** - Subscription management and quiet hours - **Delivery Tracking** - Complete delivery status and error tracking

## 🏗️ Database Schema

### Core Tables

-- Notifications: Core notification records  
notifications (id, user\_id, title, message, type, read, read\_at, module,   
 event\_type, related\_id, related\_type, metadata, expires\_at,   
 priority, created\_at, updated\_at)  
  
-- Notification Channels: Multi-channel delivery configuration  
notification\_channels (id, name, channel\_type, config, is\_active, description,  
 rate\_limit\_per\_minute, rate\_limit\_per\_hour, retry\_attempts,  
 retry\_delay, created\_by, created\_at, updated\_at)  
  
-- Notification Templates: Template-based message system  
notification\_templates (id, module, event\_type, name, subject, body, format,  
 is\_active, variables, conditions, version, parent\_id,  
 created\_by, created\_at, updated\_at)  
  
-- Notification Subscriptions: User preference management  
notification\_subscriptions (id, user\_id, module, event\_type, channel\_id,  
 is\_active, preferences, frequency, quiet\_hours,  
 created\_at, updated\_at)  
  
-- Notification Deliveries: Delivery tracking and analytics  
notification\_deliveries (id, notification\_id, channel\_id, template\_id, event\_type,  
 recipient, content, status, sent\_at, delivered\_at, failed\_at,  
 error\_message, error\_code, retry\_count, next\_retry\_at,  
 metadata, external\_id, cost, created\_at, updated\_at)

### Relationships

Users ←→ Notifications (user\_id)  
Users ←→ NotificationSubscriptions (user\_id)  
NotificationChannels ←→ NotificationSubscriptions (channel\_id)  
NotificationChannels ←→ NotificationDeliveries (channel\_id)  
NotificationTemplates ←→ NotificationDeliveries (template\_id)  
Notifications ←→ NotificationDeliveries (notification\_id)

## 📢 Notification Types

### Core Types

const NOTIFICATION\_TYPES = [  
 'info', // General information and updates  
 'success', // Success confirmations and completions  
 'warning', // Warning messages and alerts  
 'error', // Error notifications and failures  
 'alert', // Important alerts requiring attention  
 'reminder', // Reminders and scheduled notifications  
 'system', // System-related notifications  
 'security' // Security alerts and notifications  
];

### Priority Levels

const PRIORITY\_LEVELS = {  
 1: 'low', // Low priority, can be batched  
 2: 'medium', // Normal priority, standard delivery  
 3: 'high', // High priority, expedited delivery  
 4: 'urgent' // Urgent, immediate delivery required  
};

### Notification Structure

const notification = {  
 userId: 123,  
 title: 'Security Alert: Suspicious Login Detected',  
 message: 'A suspicious login attempt was detected on your account.',  
 type: 'security',  
 module: 'security',  
 eventType: 'login\_attempt',  
 relatedId: 12345,  
 relatedType: 'login\_attempt',  
 metadata: {  
 ip\_address: '192.168.1.100',  
 location: 'New York, NY',  
 device: 'Chrome on Windows 10',  
 risk\_score: 85  
 },  
 priority: 3, // High priority  
 expiresAt: new Date(Date.now() + 7 \* 24 \* 60 \* 60 \* 1000) // 7 days  
};

## 📡 Channel System

### Channel Types

const CHANNEL\_TYPES = [  
 'email', // SMTP-based email notifications  
 'sms', // SMS gateway integration  
 'push', // Push notifications for mobile/web  
 'webhook', // HTTP webhook notifications  
 'slack', // Slack channel notifications  
 'teams', // Microsoft Teams notifications  
 'discord', // Discord server notifications  
 'in\_app' // In-application notifications  
];

### Channel Configuration

// Email channel example  
const emailChannel = {  
 name: 'Email Security Alerts',  
 channelType: 'email',  
 config: {  
 smtp\_host: 'smtp.gmail.com',  
 smtp\_port: 587,  
 smtp\_user: 'alerts@company.com',  
 smtp\_secure: true,  
 from\_name: 'Security Team',  
 from\_email: 'security@company.com'  
 },  
 rateLimitPerMinute: 30,  
 rateLimitPerHour: 500,  
 retryAttempts: 3,  
 retryDelay: 300,  
 isActive: true  
};  
  
// Slack channel example  
const slackChannel = {  
 name: 'Slack Security Channel',  
 channelType: 'slack',  
 config: {  
 webhook\_url: 'https://hooks.slack.com/services/...',  
 channel: '#security-alerts',  
 username: 'SecurityBot',  
 icon\_emoji: ':warning:'  
 },  
 rateLimitPerMinute: 60,  
 rateLimitPerHour: 1000,  
 retryAttempts: 2,  
 retryDelay: 180  
};

### Rate Limiting

// Channel rate limiting configuration  
const rateLimits = {  
 email: {  
 perMinute: 30, // 30 emails per minute  
 perHour: 500, // 500 emails per hour  
 burstLimit: 10 // Allow 10 immediate sends  
 },  
 sms: {  
 perMinute: 10, // 10 SMS per minute  
 perHour: 100, // 100 SMS per hour  
 burstLimit: 3 // Allow 3 immediate sends  
 }  
};

## 📝 Template System

### Template Formats

const TEMPLATE\_FORMATS = [  
 'html', // Rich HTML templates with styling  
 'text', // Plain text templates  
 'markdown', // Markdown format templates  
 'json' // Structured JSON templates  
];

### Template Structure

const template = {  
 module: 'security',  
 eventType: 'login\_attempt',  
 name: 'Security Alert - Suspicious Login',  
 subject: 'Security Alert: Suspicious login attempt detected',  
 body: `  
 <html>  
 <body>  
 <h2>Security Alert</h2>  
 <p>Hello {{user\_name}},</p>  
 <p>We detected a suspicious login attempt on your account:</p>  
 <ul>  
 <li><strong>Time:</strong> {{timestamp}}</li>  
 <li><strong>IP Address:</strong> {{ip\_address}}</li>  
 <li><strong>Location:</strong> {{location}}</li>  
 <li><strong>Device:</strong> {{device}}</li>  
 </ul>  
 <p>If this was not you, please contact security immediately.</p>  
 </body>  
 </html>  
 `,  
 format: 'html',  
 variables: ['user\_name', 'timestamp', 'ip\_address', 'location', 'device'],  
 conditions: {  
 priority: 'high',  
 send\_immediately: true  
 },  
 version: 1,  
 isActive: true  
};

### Variable Substitution

// Template variables are replaced with actual values  
const variables = {  
 user\_name: 'John Doe',  
 timestamp: '2024-01-15 14:30:00',  
 ip\_address: '192.168.1.100',  
 location: 'New York, NY',  
 device: 'Chrome on Windows 10'  
};  
  
// Result: "Hello John Doe, we detected..."

### Conditional Logic

// Templates can include conditional logic  
const conditionalTemplate = {  
 body: `  
 {{#if high\_priority}}  
 <div style="color: red; font-weight: bold;">URGENT</div>  
 {{/if}}  
   
 <p>{{message}}</p>  
   
 {{#if show\_details}}  
 <ul>  
 {{#each details}}  
 <li>{{this}}</li>  
 {{/each}}  
 </ul>  
 {{/if}}  
 `,  
 conditions: {  
 high\_priority: '{{priority}} >= 3',  
 show\_details: '{{user\_role}} === "admin"'  
 }  
};

## 👤 User Preferences

### Subscription Management

const subscription = {  
 userId: 123,  
 module: 'security',  
 eventType: 'login\_attempt',  
 channelId: 1, // Email channel  
 isActive: true,  
 preferences: {  
 digest: false, // Send individual notifications  
 priority\_filter: 2, // Only medium priority and above  
 keywords: ['critical', 'urgent']  
 },  
 frequency: 'immediate', // immediate, hourly, daily, weekly  
 quietHours: {  
 enabled: true,  
 start: '22:00',  
 end: '08:00',  
 timezone: 'America/New\_York'  
 }  
};

### Frequency Options

const FREQUENCY\_OPTIONS = [  
 'immediate', // Send immediately  
 'hourly', // Batch and send hourly  
 'daily', // Daily digest  
 'weekly' // Weekly summary  
];

## 📊 Delivery Tracking

### Delivery Status

const DELIVERY\_STATUSES = [  
 'pending', // Queued for delivery  
 'sent', // Sent to channel  
 'delivered', // Confirmed delivery  
 'failed', // Delivery failed  
 'bounced', // Message bounced  
 'cancelled' // Delivery cancelled  
];

### Delivery Record

const delivery = {  
 notificationId: 456,  
 channelId: 1,  
 templateId: 789,  
 eventType: 'login\_attempt',  
 recipient: {  
 email: 'user@example.com',  
 name: 'John Doe'  
 },  
 content: {  
 subject: 'Security Alert',  
 body: 'Rendered template content...',  
 format: 'html'  
 },  
 status: 'delivered',  
 sentAt: '2024-01-15T14:30:00Z',  
 deliveredAt: '2024-01-15T14:30:05Z',  
 retryCount: 0,  
 cost: 5, // Cost in cents  
 metadata: {  
 messageId: 'msg\_123456',  
 provider: 'sendgrid'  
 }  
};

## 🚀 API Endpoints

### Core Notifications

// Basic notification operations  
POST /api/v1/notifications // Create notification  
GET /api/v1/notifications // Get user notifications  
PATCH /api/v1/notifications/:id/read // Mark as read  
PATCH /api/v1/notifications/read-all // Mark all as read  
DELETE /api/v1/notifications/:id // Delete notification  
GET /api/v1/notifications/stats // Get statistics

### Notification Channels

// Channel management (admin only)  
POST /api/v1/notifications/channels // Create channel  
GET /api/v1/notifications/channels // Get all channels  
PUT /api/v1/notifications/channels/:id // Update channel  
DELETE /api/v1/notifications/channels/:id // Delete channel

### Notification Templates

// Template management (admin only)  
POST /api/v1/notifications/templates // Create template  
GET /api/v1/notifications/templates // Get all templates  
GET /api/v1/notifications/templates/:module/:event // Get by module/event  
PUT /api/v1/notifications/templates/:id // Update template  
DELETE /api/v1/notifications/templates/:id // Delete template

## 🛠️ Usage Examples

### Creating a Notification

const notificationData = {  
 userId: 123,  
 title: 'System Maintenance Scheduled',  
 message: 'System maintenance is scheduled for tonight from 2:00 AM to 4:00 AM EST.',  
 type: 'system',  
 module: 'system',  
 eventType: 'maintenance',  
 priority: 2,  
 metadata: {  
 maintenance\_window: '2024-01-16 02:00:00 - 2024-01-16 04:00:00',  
 affected\_services: ['dashboard', 'api', 'reports']  
 }  
};  
  
const notification = await notificationService.createNotification(notificationData);

### Setting Up a Channel

const channelData = {  
 name: 'Emergency SMS Alerts',  
 channelType: 'sms',  
 config: {  
 provider: 'twilio',  
 account\_sid: 'AC123...',  
 auth\_token: 'token123...',  
 from\_number: '+1234567890'  
 },  
 rateLimitPerMinute: 10,  
 rateLimitPerHour: 100,  
 retryAttempts: 3,  
 retryDelay: 300,  
 isActive: true  
};  
  
const channel = await notificationService.createChannel(channelData, adminUserId);

### Creating a Template

const templateData = {  
 module: 'system',  
 eventType: 'maintenance',  
 name: 'System Maintenance Notification',  
 subject: 'Scheduled Maintenance: {{service\_name}}',  
 body: `  
 <h2>Scheduled Maintenance Notice</h2>  
 <p>Dear {{user\_name}},</p>  
 <p>We will be performing scheduled maintenance on {{service\_name}}:</p>  
 <ul>  
 <li><strong>Start:</strong> {{start\_time}}</li>  
 <li><strong>End:</strong> {{end\_time}}</li>  
 <li><strong>Duration:</strong> {{duration}}</li>  
 </ul>  
 <p>During this time, the following services may be unavailable:</p>  
 <ul>  
 {{#each affected\_services}}  
 <li>{{this}}</li>  
 {{/each}}  
 </ul>  
 <p>We apologize for any inconvenience.</p>  
 `,  
 format: 'html',  
 variables: ['user\_name', 'service\_name', 'start\_time', 'end\_time', 'duration', 'affected\_services'],  
 isActive: true  
};  
  
const template = await notificationService.createTemplate(templateData, adminUserId);

### Managing User Preferences

// Get user notifications with filtering  
const notifications = await notificationService.getUserNotifications(userId, {  
 type: 'security',  
 read: false,  
 priority: 3,  
 startDate: '2024-01-01',  
 endDate: '2024-01-31'  
}, {  
 page: 1,  
 limit: 20,  
 sortBy: 'createdAt',  
 sortOrder: 'desc'  
});  
  
// Mark notifications as read  
await notificationService.markAsRead(notificationId, userId);  
  
// Mark all security notifications as read  
await notificationService.markAllAsRead(userId, { type: 'security' });

### Getting Analytics

// Get notification statistics  
const stats = await notificationService.getUserNotificationStats(userId);  
  
console.log('Notification Statistics:');  
console.log(`Total: ${stats.overview.total}`);  
console.log(`Unread: ${stats.overview.unread}`);  
console.log(`High Priority: ${stats.overview.highPriority}`);  
  
// Statistics by type  
stats.byType.forEach(type => {  
 console.log(`${type.type}: ${type.count} (${type.unreadCount} unread)`);  
});  
  
// Statistics by module  
stats.byModule.forEach(module => {  
 console.log(`${module.module}: ${module.count} (${module.unreadCount} unread)`);  
});