n8n Workflow Automation Integration

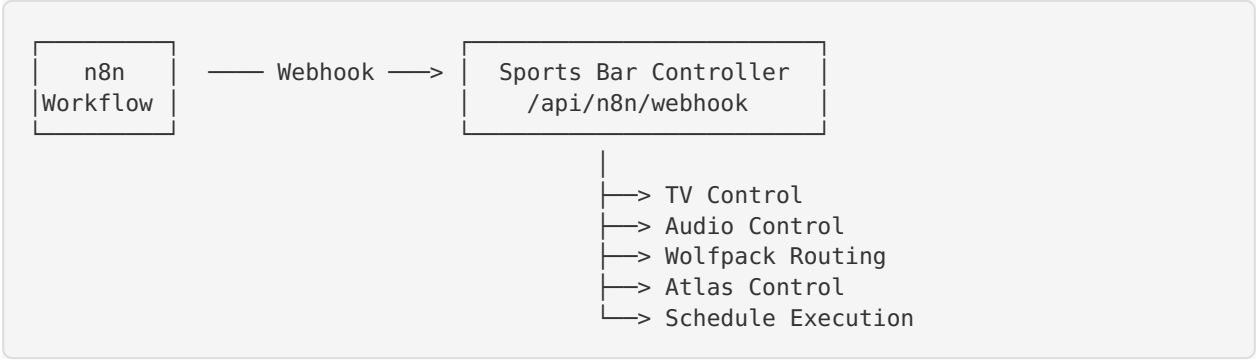
Overview

The Sports Bar TV Controller integrates with n8n (https://n8n.io), an open-source workflow automation platform, enabling automated control of TVs, audio zones, and video routing based on schedules, events, or external triggers.

Features

- **Webhook-based Integration**: Receive commands from n8n workflows via webhooks
- **Action Logging**: Track all n8n-triggered actions with detailed logs
- **Multiple Action Types**: Control TVs, audio, Wolfpack routing, Atlas processor, and more
- **Secure Communication**: Token-based authentication for webhook endpoints
- **Comprehensive Response**: Detailed success/failure responses for workflow decision-making

Architecture



Setup

1. Configure Webhook Token

Set an environment variable for webhook security:

```
# .env.local
N8N_WEBHOOK_TOKEN=your-secure-token-here
NEXT_PUBLIC_APP_URL=http://your-server:3001
```

2. Install n8n (Optional - Self-Hosted)

```
# Using Docker
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  -v ~/.n8n:/home/node/.n8n \
  n8nio/n8n

# Using npm
npm install n8n -g
n8n
```

Access n8n at: `http://localhost:5678`

3. Run Database Migration

Generate and apply the schema changes:

```
npm run db:generate
npm run db:push
```

Webhook Endpoint

URL

```
POST http://your-server:3001/api/n8n/webhook
```

Authentication

Include the webhook token in the Authorization header:

```
Authorization: Bearer your-secure-token-here
```

Request Format

```
{
  "action": "control_tv",
  "data": {
    "outputNumber": 1,
    "action": "power_on"
  },
  "workflowId": "workflow-123",
  "executionId": "exec-456",
  "metadata": {
    "source": "scheduled_automation"
  }
}
```

Response Format

```
{
  "success": true,
  "action": "control_tv",
  "result": {
    "message": "TV powered on successfully"
  },
  "duration": 1234,
  "error": null
}
```

Supported Actions

1. Control TV (`control_tv`)

Power control, input selection, and channel changes for TVs.

Request:

```
{
  "action": "control_tv",
  "data": {
    "outputNumber": 1,
    "action": "power_on",  // or "power_off", "set_input", "set_channel"
    "value": 3             // Optional: input/channel number
  }
}
```

Example Workflow:
- Schedule TVs to turn on at opening time
- Automatically switch inputs based on events
- Turn off TVs at closing time

2. Control Audio (`control_audio`)

Manage audio zones, volume, and source routing.

Request:

```
{
  "action": "control_audio",
  "data": {
    "zoneNumber": 1,
    "action": "set_volume",  // or "mute", "unmute", "set_source"
    "value": 50              // Volume level 0-100 or source number
  }
}
```

Example Workflow:
- Adjust volume for different times of day
- Mute zones during special announcements
- Switch audio sources for sports events

3. Route Wolfpack (`route_wolfpack`)

Route Wolfpack video matrix inputs to outputs for Atlas audio integration.

Request:

```
{
  "action": "route_wolfpack",
  "data": {
    "wolfpackInputNumber": 1,
    "matrixOutputNumber": 3
  }
}
```

Example Workflow:
- Automatically route video inputs based on scheduled content
- Switch to broadcast TV for major sports events
- Route streaming device to specific zones

4. Execute Schedule (`execute_schedule`)

Trigger a predefined schedule configuration.

Request:

```
{
  "action": "execute_schedule",
  "data": {
    "scheduleId": "schedule-uuid-here"
  }
}
```

Example Workflow:
- Trigger complex multi-step schedules
- Execute “Game Day” configuration
- Run “Opening” or “Closing” routines

5. Control Atlas (`control_atlas`)

Direct control of Atlas processor parameters.

Request:

```
{
  "action": "control_atlas",
  "data": {
    "command": "route_to_zone",
    "parameters": {
      "matrixInputNumber": 1,
      "zoneNumber": 2
    }
  }
}
```

Example Workflow:
- Advanced audio routing

- Dynamic audio processing adjustments
- Zone-specific audio configurations

6. Health Check (`health_check`)

Verify the webhook endpoint is operational.

Request:

```
{
  "action": "health_check",
  "data": {}
}
```

Example n8n Workflows

Example 1: Daily Opening Routine

1. **Schedule Trigger** - 10:00 AM daily
2. **HTTP Request** - Power on all TVs
   `POST /api/n8n/webhook`
   `Body: {"action": "control_tv", "data": {"outputNumber": 1, "action": "power_on"}}`
3. **HTTP Request** - Set default audio levels
4. **HTTP Request** - Route default video inputs

Example 2: Game Day Automation

1. **Webhook Trigger** - External sports API notifies of game start
2. **HTTP Request** - Execute game day schedule
   `POST /api/n8n/webhook`
   `Body: {"action": "execute_schedule", "data": {"scheduleId": "game-day-uuid"}}`
3. **HTTP Request** - Route ESPN to main TVs
4. **HTTP Request** - Increase audio volume in main zones

Example 3: Conditional Audio Routing

1. **Schedule Trigger** - Check every hour
2. **HTTP Request** - Get current time and occupancy
3. **IF Node** - If busy hours (Friday/Saturday evening)
   - **Then**: Increase volume, route live music
   - **Else**: Lower volume, route background music

Creating n8n Workflows

Basic Workflow Template

1. **Add Webhook/Schedule Trigger**
   - For webhooks: Create a new webhook trigger node
   - For schedules: Use “Schedule Trigger” node with cron expression

2. **Add HTTP Request Node**
   - Method: POST

- URL: `http://your-server:3001/api/n8n/webhook`
- Headers:
  - `Content-Type`: `application/json`
  - `Authorization`: `Bearer your-token`
  - Body: JSON with action and data

3. **Add Response Handling**
   - Use IF node to check `success` field
   - Log successes and failures
   - Send notifications on errors (Slack, email, etc.)

4. **Save and Activate Workflow**

Monitoring and Logs

View Webhook Logs

All n8n webhook executions are logged to the database:

```
SELECT * FROM N8nWebhookLog
ORDER BY createdAt DESC
LIMIT 50;
```

Log Fields

- `id`: Unique log ID
- `action`: Action type that was executed
- `workflowId`: n8n workflow ID
- `executionId`: n8n execution ID
- `payload`: Full request payload
- `response`: Response sent back to n8n
- `status`: success, failed, or error
- `errorMessage`: Error details if failed
- `duration`: Execution time in milliseconds
- `metadata`: Additional context
- `createdAt`: Timestamp

Workflow Configurations

Store and manage n8n workflow configurations:

```
SELECT * FROM N8nWorkflowConfig
WHERE isActive = 1;
```

Security Best Practices

1. **Use Strong Webhook Tokens**
   - Generate secure random tokens
   - Rotate tokens periodically
   - Never commit tokens to version control

2. **Restrict Network Access**
   - Use firewall rules to limit webhook access
   - Consider VPN for production deployments
   - Use HTTPS in production

3. **Validate Inputs**
   - The webhook validates all required fields
   - Invalid requests return 400 Bad Request
   - Unauthorized requests return 401 Unauthorized

4. **Monitor Logs**
   - Regularly review webhook logs for suspicious activity
   - Set up alerts for failed authentication attempts
   - Track unusual action patterns

Troubleshooting

Webhook Returns 401 Unauthorized

Cause: Invalid or missing authorization token

Solution:
- Check `N8N_WEBHOOK_TOKEN` environment variable
- Verify Authorization header format: `Bearer your-token`
- Ensure token matches in both n8n and application

Action Fails with 500 Error

Cause: Internal API error or invalid parameters

Solution:
- Check webhook logs for error details
- Verify data parameters match action requirements
- Check application logs for stack traces
- Ensure target systems (Atlas, TVs, etc.) are accessible

Workflow Not Executing

Cause: n8n workflow configuration issue

Solution:
- Verify workflow is activated in n8n
- Check n8n execution logs
- Test webhook manually with curl:

```bash
curl -X POST http://your-server:3001/api/n8n/webhook \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer your-token" \
  -d '{"action":"health_check","data":{}}'
```

Timeout Errors

Cause: Long-running operations or network issues

Solution:
- Increase n8n timeout settings
- Check network connectivity
- Consider async operations for complex workflows
- Split large workflows into smaller steps

API Reference

GET /api/n8n/webhook

Get webhook endpoint information and supported actions.

Response:

```
{
  "message": "n8n Webhook Endpoint",
  "status": "active",
  "supportedActions": [
    "control_tv",
    "control_audio",
    "route_wolfpack",
    "execute_schedule",
    "control_atlas",
    "health_check"
  ],
  "documentation": "/docs/n8n-integration"
}
```

POST /api/n8n/webhook

Execute an action via n8n webhook.

Required Headers:
- `Content-Type: application/json`
- `Authorization: Bearer <token>` (if N8N_WEBHOOK_TOKEN is set)

Request Body:
- `action` (string, required): Action to execute
- `data` (object, required): Action-specific parameters
- `workflowId` (string, optional): n8n workflow ID
- `executionId` (string, optional): n8n execution ID
- `metadata` (object, optional): Additional context

Response Codes:
- `200 OK` : Action executed successfully
- `400 Bad Request` : Invalid request format or missing required fields
- `401 Unauthorized` : Missing or invalid authorization token
- `500 Internal Server Error` : Server error during execution

Advanced Use Cases

Multi-Zone Audio Synchronization

Use n8n to synchronize audio across multiple zones:

1. Trigger on specific event (e.g., live sports broadcast starts)
2. Parallel HTTP requests to set same source for all zones
3. Adjust volumes based on zone characteristics
4. Monitor for failures and send alerts

Dynamic Content Routing

Route video content based on external factors:

1. Poll sports API for game status
2. When game starts, route appropriate input to TVs
3. Adjust audio routing to match video
4. Return to default content when game ends

Automated Maintenance

Schedule regular maintenance tasks:

1. Nightly: Turn off all systems
2. Weekly: Test all routing combinations
3. Monthly: Generate usage reports
4. Quarterly: Backup configurations

Support and Resources

- **n8n Documentation**: https://docs.n8n.io
- **n8n Community**: https://community.n8n.io
- **Sports Bar Controller Docs**: `/docs/`
- **API Documentation**: `/api/`

Contributing

To extend n8n integration:

1. Add new action handlers in `/src/app/api/n8n/webhook/route.ts`
2. Update documentation with new action examples
3. Add tests for new actions
4. Update the supportedActions array

License

This integration follows the same license as the Sports Bar TV Controller application.