Cron Schedule Configuration

This document outlines the recommended schedules for all cron jobs in the Call a Star platform.

Overview

The platform uses 5 cron jobs for automated maintenance, payment processing, and user notifications. Each cron job is secured with the `CRON_SECRET` environment variable.

Cron Jobs and Recommended Schedules

1. cleanup-logs

Endpoint: `/api/cron/cleanup-logs`
Schedule: `0 2 * * *` (Daily at 2:00 AM UTC)
Frequency: Once per day

Purpose:
Cleans up old log entries from the database based on retention policies:
- INFO logs: 30 days
- WARNING logs: 60 days
- ERROR logs: 90 days
- CRITICAL logs: Unlimited (never deleted)

Why this schedule:
Running at 2 AM UTC ensures minimal impact on users and database performance during low-traffic hours.

---

2. process-payouts

Endpoint: `/api/cron/process-payouts`
Schedule: `0 * * * *` (Every hour at minute 0)
Frequency: Every hour

Purpose:
Processes manual payout requests for creators. Checks eligibility, creates payout records, and sends notifications to admins for approval.

Why this schedule:
Hourly execution ensures timely processing of payout requests without overwhelming the system.

---

3. process-automatic-payouts

Endpoint: `/api/cron/process-automatic-payouts`
Schedule: `0 1 * * *` (Daily at 1:00 AM UTC)
Frequency: Once per day

Purpose:
Processes automatic payouts for creators with DAILY or WEEKLY payout schedules. Creates Stripe payouts for eligible creators based on their available balance.

Why this schedule:
Running at 1 AM UTC (before log cleanup) ensures creators receive their payouts promptly while avoiding peak hours.

---

4. cleanup-daily-rooms

Endpoint: `/api/cron/cleanup-daily-rooms`
Schedule: `0 * * * *` (Every hour at minute 0)
Frequency: Every hour

Purpose:
Cleans up Daily.io video rooms for completed or cancelled bookings. Verifies room existence before deletion and updates booking records.

Why this schedule:
Hourly cleanup ensures rooms are deleted promptly after calls end, preventing unnecessary Daily.io charges and maintaining a clean room list.

---

5. send-booking-reminders

Endpoint: `/api/cron/send-booking-reminders`
Schedule: `*/5 * * * *` (Every 5 minutes)
Frequency: Every 5 minutes

Purpose:
Sends email reminders to clients and creators 15 minutes before their scheduled calls. Emails are in English and include links to the platform booking page.

Why this schedule:
5-minute intervals ensure reminders are sent within the 15-20 minute window before calls, maximizing the chance of users joining on time.

Vercel Cron Configuration

To configure these cron jobs on Vercel, add the following to your `vercel.json` file:

```
{
  "crons": [
    {
      "path": "/api/cron/cleanup-logs",
      "schedule": "0 2 * * *"
    },
    {
      "path": "/api/cron/process-payouts",
      "schedule": "0 * * * *"
    },
    {
      "path": "/api/cron/process-automatic-payouts",
      "schedule": "0 1 * * *"
    },
    {
      "path": "/api/cron/cleanup-daily-rooms",
      "schedule": "0 * * * *"
    },
    {
      "path": "/api/cron/send-booking-reminders",
      "schedule": "*/5 * * * *"
    }
  ]
}
```

GitHub Actions Alternative

If you prefer using GitHub Actions instead of Vercel Cron, create the following workflow files:

`.github/workflows/cron-cleanup-logs.yml`

```
name: Cleanup Logs Cron

on:
  schedule:
    - cron: '0 2 * * *'  # Daily at 2 AM UTC
  workflow_dispatch:

jobs:
  cleanup-logs:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger cleanup-logs cron
        run: |
          curl -X POST "${{ secrets.APP_URL }}/api/cron/cleanup-logs" \
            -H "Authorization: Bearer ${{ secrets.CRON_SECRET }}"
```

`.github/workflows/cron-process-payouts.yml`

```yaml
name: Process Payouts Cron

on:
  schedule:
    - cron: '0 * * * *'  # Every hour
  workflow_dispatch:

jobs:
  process-payouts:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger process-payouts cron
        run: |
          curl -X POST "${{ secrets.APP_URL }}/api/cron/process-payouts" \
            -H "x-cron-secret: ${{ secrets.CRON_SECRET }}"
```

`.github/workflows/cron-process-automatic-payouts.yml`

```yaml
name: Process Automatic Payouts Cron

on:
  schedule:
    - cron: '0 1 * * *'  # Daily at 1 AM UTC
  workflow_dispatch:

jobs:
  process-automatic-payouts:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger process-automatic-payouts cron
        run: |
          curl -X POST "${{ secrets.APP_URL }}/api/cron/process-automatic-payouts" \
            -H "Authorization: Bearer ${{ secrets.CRON_SECRET }}"
```

`.github/workflows/cron-cleanup-daily-rooms.yml`

```yaml
name: Cleanup Daily Rooms Cron

on:
  schedule:
    - cron: '0 * * * *'  # Every hour
  workflow_dispatch:

jobs:
  cleanup-daily-rooms:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger cleanup-daily-rooms cron
        run: |
          curl -X POST "${{ secrets.APP_URL }}/api/cron/cleanup-daily-rooms" \
            -H "Authorization: Bearer ${{ secrets.CRON_SECRET }}"
```

`.github/workflows/cron-send-booking-reminders.yml`

```yaml
name: Send Booking Reminders Cron

on:
  schedule:
    - cron: '*/5 * * * *'  # Every 5 minutes
  workflow_dispatch:

jobs:
  send-booking-reminders:
    runs-on: ubuntu-latest
    steps:
      - name: Trigger send-booking-reminders cron
        run: |
          curl -X POST "${{ secrets.APP_URL }}/api/cron/send-booking-reminders" \
            -H "Authorization: Bearer ${{ secrets.CRON_SECRET }}"
```

Security

All cron endpoints are protected by the `CRON_SECRET` environment variable. Requests must include the secret in the `Authorization` header:

```
Authorization: Bearer <CRON_SECRET>
```

Or for some endpoints, in the `x-cron-secret` header:

```
x-cron-secret: <CRON_SECRET>
```

Make sure to set the `CRON_SECRET` environment variable in your deployment environment.

Monitoring

All cron jobs log their execution to the database using the logging system (`lib/logger.ts`). You can monitor cron execution by:

1. Checking the `Log` table in the database
2. Filtering by `type = 'CRON_RUN'` for successful executions
3. Filtering by `type = 'CRON_ERROR'` for failed executions
4. Using the admin dashboard (if available) to view recent logs

Each log entry includes:
- Cron type
- Number of items processed
- Duration (in milliseconds)
- Success/error status
- Detailed context information

Testing

You can manually trigger any cron job by making a POST request to its endpoint with the correct authorization:

```
curl -X POST "https://your-domain.com/api/cron/cleanup-logs" \
  -H "Authorization: Bearer your-cron-secret"
```

Or test the GET endpoint to verify the cron is configured correctly:

```
curl "https://your-domain.com/api/cron/cleanup-logs"
```

Adjusting Schedules

If you need to adjust the schedules:

1. Update the cron expression in `vercel.json` or your GitHub Actions workflows
2. Test the new schedule in a staging environment first
3. Monitor the logs after deployment to ensure proper execution
4. Consider timezone differences when setting schedules (all times are UTC)

Cron Expression Reference

- `0 2 * * *` - Daily at 2:00 AM
- `0 1 * * *` - Daily at 1:00 AM
- `0 * * * *` - Every hour
- `*/5 * * * *` - Every 5 minutes
- `0 */6 * * *` - Every 6 hours
- `0 0 * * 0` - Weekly on Sundays at midnight

Use crontab.guru (https://crontab.guru/) to help create and understand cron expressions.