NFHS Network API Integration Guide

Overview

This document explains how to integrate with the NFHS Network API once it becomes available. Currently, NFHS Network does **NOT** have a public API, but this framework has been prepared to make integration easier when API access is granted.

Current Status

- ⚠️ **No Public API Available**: NFHS Network does not currently offer a public API
- ✅ **Framework Ready**: Placeholder code structure is in place at `src/lib/streaming/api-integrations/nfhs-api.ts`
- ✅ **App Detection Works**: The Fire TV app detection can identify when NFHS Network app is installed
- ✅ **App Launching Works**: The system can launch the NFHS Network app on Fire TV devices

Requesting API Access

Contact Information

To request API access from NFHS Network, use the following contact methods:

1. **Email Support**: support@nfhsnetwork.com
2. **Website Contact Form**: https://www.nfhsnetwork.com/contact
3. **Phone**: (404) 885-4786
4. **Business Development**: partnerships@nfhsnetwork.com

What to Include in Your Request

When contacting NFHS for API access, include the following information:

```
Subject: API Access Request for Sports Bar TV Controller Integration

Dear NFHS Network Team,

I am writing to request API access for integration with our Sports Bar TV Controller s
ystem.

Business Use Case:
- We are developing a TV control system for sports bars and restaurants
- The system helps manage multiple Fire TV devices displaying sports content
- We want to provide schedule data and automated app launching for NFHS events

Technical Requirements:
- Read-only access to event schedules and live stream information
- Ability to search for schools and events
- Event metadata (date, time, sport, teams, etc.)
- Deep linking support for launching to specific events

Expected Volume:
- [Your expected API call volume]
- [Number of venues/devices]

Contact Information:
- Name: [Your Name]
- Company: [Your Company]
- Email: [Your Email]
- Phone: [Your Phone]

Thank you for considering our request.
```

Data Requirements

When the API becomes available, we will need access to the following data:

1. Event Schedule Data

```
{
  id: string                      // Unique event identifier
  title: string                   // Event title (e.g., "Lions vs Tigers")
  schoolName: string              // Primary school name
  opposingSchool?: string         // Opposing team/school
  sport: string                   // Sport type (football, basketball, etc.)
  date: string                    // Event date (ISO 8601)
  startTime: string               // Start time (ISO 8601)
  endTime?: string                // End time (ISO 8601)
  isLive: boolean                 // Whether event is currently live
  streamUrl?: string              // URL to stream (if available via API)
  thumbnailUrl?: string           // Event thumbnail/poster image
  state: string                   // US State
  league?: string                 // League or conference name
}
```

2. School Information

```
{
  id: string                    // Unique school identifier
  name: string                  // School name
  state: string                 // US State
  city: string                  // City
  mascot?: string               // Team mascot
  colors?: string[]             // School colors
}
```

3. Live Event Data

```
{
  eventId: string               // Reference to event
  isLive: boolean               // Current live status
  viewerCount?: number          // Current viewer count
  streamQuality?: string        // Available quality levels
}
```

API Endpoints (Expected)

Once the API is available, we expect the following endpoints:

GET /api/v1/events

Get upcoming events with optional filters

Query Parameters:
- `sport` - Filter by sport type
- `state` - Filter by US state
- `schoolName` - Filter by school name
- `date` - Filter by date (ISO 8601)
- `limit` - Number of results per page
- `offset` - Pagination offset

Example:

```
GET https://api.nfhsnetwork.com/v1/events?sport=football&state=GA&limit=20
```

GET /api/v1/events/live

Get currently live events

Query Parameters:
- `sport` - Filter by sport type
- `state` - Filter by US state

GET /api/v1/events/{eventId}

Get detailed information for a specific event

GET /api/v1/schools

Search for schools

Query Parameters:
- `query` - Search query
- `state` - Filter by state

GET /api/v1/schools/{schoolId}/schedule

Get schedule for a specific school

Query Parameters:
- `sport` - Filter by sport
- `startDate` - Schedule start date
- `endDate` - Schedule end date

Integration Steps

When API access is granted, follow these steps:

1. Obtain API Credentials

Get your API key and secret from NFHS Network:
- API Key
- API Secret (if required)
- Base URL for API endpoints

2. Configure Environment Variables

Add the following to your `.env.local` file:

```
# NFHS Network API Configuration
NFHS_API_KEY=your_api_key_here
NFHS_API_SECRET=your_api_secret_here
NFHS_API_BASE_URL=https://api.nfhsnetwork.com/v1
```

3. Update the API Client

Edit `src/lib/streaming/api-integrations/nfhs-api.ts`:

1. Remove the `// TODO:` comments
2. Implement actual fetch calls using the base URL and credentials
3. Add proper error handling
4. Implement rate limiting if required by NFHS

Example implementation:

```typescript
public async getUpcomingEvents(params?: {
  sport?: string
  state?: string
  schoolName?: string
  date?: string
  limit?: number
}): Promise<NFHSSchedule> {
  const queryParams = new URLSearchParams()
  if (params?.sport) queryParams.append('sport', params.sport)
  if (params?.state) queryParams.append('state', params.state)
  if (params?.schoolName) queryParams.append('schoolName', params.schoolName)
  if (params?.date) queryParams.append('date', params.date)
  if (params?.limit) queryParams.append('limit', params.limit.toString())

  const response = await fetch(
    `${this.config.baseUrl}/events?${queryParams.toString()}`,
    {
      method: 'GET',
      headers: {
        'Authorization': `Bearer ${this.config.apiKey}`,
        'Content-Type': 'application/json'
      }
    }
  )

  if (!response.ok) {
    throw new Error(`NFHS API error: ${response.status} ${response.statusText}`)
  }

  return await response.json()
}
```

4. Create API Route Handlers

Create Next.js API routes to expose NFHS data to your frontend:

File: `src/app/api/streaming/nfhs/events/route.ts`

```
import { NextRequest, NextResponse } from 'next/server'
import { nfhsApi } from '@/lib/streaming/api-integrations/nfhs-api'

export async function GET(request: NextRequest) {
  try {
    const searchParams = request.nextUrl.searchParams
    const sport = searchParams.get('sport') || undefined
    const state = searchParams.get('state') || undefined
    const limit = parseInt(searchParams.get('limit') || '20')

    const schedule = await nfhsApi.getUpcomingEvents({
      sport,
      state,
      limit
    })

    return NextResponse.json(schedule)
  } catch (error) {
    console.error('Error fetching NFHS events:', error)
    return NextResponse.json(
      { error: 'Failed to fetch events' },
      { status: 500 }
    )
  }
}
```

5. Update UI Components

Add UI components to display NFHS schedule data and launch events.

See `src/components/streaming/NFHSScheduleViewer.tsx` (to be created)

6. Test Integration

1. Test API calls with Postman or curl
2. Verify data is being fetched correctly
3. Test app launching with deep links
4. Verify error handling for API failures
5. Test with actual Fire TV devices

Deep Linking

NFHS Network app supports deep linking to specific events:

Format: `nfhs://event/{eventId}`

Example: `nfhs://event/abc123xyz`

To launch an event:

```
import { streamingManager } from '@/services/streaming-service-manager'

await streamingManager.launchApp(
  deviceId,
  ipAddress,
  'nfhs-network',
  {
    deepLink: 'nfhs://event/abc123xyz'
  }
)
```

Current Workarounds

Until the API is available, you can:

1. **Manual Schedule Entry**: Allow users to manually enter NFHS event information
2. **Web Scraping** (with caution): Consider scraping the NFHS website schedule pages
   - ⚠️ Check NFHS Terms of Service before implementing
   - ⚠️ Be respectful of their servers with rate limiting
3. **RSS Feeds**: Check if NFHS provides any RSS feeds for schedules
4. **Basic App Launching**: Use the existing app detection and launching without schedule data

Alternative Data Sources

While waiting for NFHS API access, consider these alternatives:

1. **MaxPreps API**: High school sports scores and schedules
2. **School District Websites**: Often have athletic calendars
3. **State Athletic Association APIs**: Some states have their own APIs
4. **Third-party Aggregators**: Services that aggregate high school sports data

Code Files Involved

When integrating the NFHS API, you’ll need to modify these files:

1. **API Client**: `src/lib/streaming/api-integrations/nfhs-api.ts`
2. **Environment Config**: `.env.local`
3. **API Routes**: `src/app/api/streaming/nfhs/` (to be created)
4. **UI Components**: `src/components/streaming/` (to be created)
5. **Type Definitions**: Update interfaces in API client as needed

Error Handling

Implement proper error handling for common scenarios:

```
try {
  const events = await nfhsApi.getUpcomingEvents()
} catch (error) {
  if (error.status === 401) {
    // API key is invalid or expired
    console.error('NFHS API authentication failed')
  } else if (error.status === 429) {
    // Rate limit exceeded
    console.error('NFHS API rate limit exceeded')
  } else if (error.status === 503) {
    // Service unavailable
    console.error('NFHS API service unavailable')
  } else {
    // Other errors
    console.error('NFHS API error:', error)
  }

  // Fallback to cached data or show error to user
}
```

Rate Limiting

Be prepared to implement rate limiting:

```
class NFHSApiClient {
  private requestQueue: Promise<any>[] = []
  private readonly MAX_REQUESTS_PER_MINUTE = 60

  private async throttleRequest<T>(fn: () => Promise<T>): Promise<T> {
    // Implement rate limiting logic
    // ...
  }
}
```

Caching Strategy

Implement caching to reduce API calls:

```
// Cache schedule data for 5 minutes
const CACHE_TTL = 5 * 60 * 1000

class NFHSApiClient {
  private cache: Map<string, { data: any; timestamp: number }> = new Map()

  private getCached<T>(key: string): T | null {
    const cached = this.cache.get(key)
    if (!cached) return null

    if (Date.now() - cached.timestamp > CACHE_TTL) {
      this.cache.delete(key)
      return null
    }

    return cached.data
  }
}
```

Support and Updates

- Check this document regularly for updates
- Monitor NFHS Network announcements for API releases
- Join developer communities for high school sports tech
- Consider reaching out to other developers who may have API access

Questions?

If you have questions about implementing the NFHS API integration:

1. Check the placeholder code in `src/lib/streaming/api-integrations/nfhs-api.ts`
2. Review the streaming manager service for usage examples
3. Contact the development team
4. Refer to NFHS Network's documentation (when available)

Last Updated: October 28, 2025

Status: Awaiting NFHS API Access

Framework Version: 1.0.0