Unified Global Cache & IR Device Setup System

Overview

This document describes the unified device setup system that was implemented to streamline the configuration of Global Cache IR controllers and IR-controlled devices (cable boxes, receivers, etc.) in the Sports Bar TV Controller application.

System Architecture

Database Schema

New Models Added

1. **IRDevice** - Represents devices controlled via IR
   - `id`: Unique identifier
   - `name`: Device name (e.g., "Cable Box 1")
   - `deviceType`: Type of device (Cable Box, Satellite, AV Receiver, etc.)
   - `brand`: Manufacturer (e.g., "DirectTV", "Dish")
   - `model`: Optional model number
   - `matrixInput`: Which matrix input the device is connected to
   - `matrixInputLabel`: Label for the matrix input
   - `irCodeSetId`: Global Cache IR Database codeset ID
   - `description`: Optional notes
   - `status`: Device status (active/inactive)
   - Relationships: Has many `GlobalCachePort` and `IRCommand`

2. **IRCommand** - Stores downloaded IR commands
   - `id`: Unique identifier
   - `deviceId`: Foreign key to IRDevice
   - `functionName`: Command name (e.g., "POWER", "CHANNEL UP")
   - `irCode`: IR code in Global Cache format
   - `hexCode`: Optional hex format code
   - `codeSetId`: Global Cache codeset ID
   - `category`: Command category (Power, Volume, Channel, etc.)
   - `description`: Optional notes

3. **IRDatabaseCredentials** - Stores Global Cache account credentials
   - `id`: Unique identifier
   - `email`: Account email
   - `password`: Encrypted password
   - `apiKey`: Current session API key
   - `isActive`: Whether this credential set is active
   - `lastLogin`: Last login timestamp
   - `dailyLimit`: Daily download limit
   - `usedToday`: Downloads used today
   - `lastReset`: Last time the daily counter was reset

Updated Models

- **GlobalCachePort** - Now links to IRDevice
- Added `irDevice` relationship field
- Links Global Cache ports to specific IR devices

Backend Services

IR Database Service (`src/lib/services/ir-database.ts`)

Handles communication with the Global Cache Cloud IR Database API.

Key Methods:
- `login(email, password)` - Authenticate with Global Cache
- `logout(apiKey)` - End session
- `getBrands()` - Get list of all brands
- `getTypes()` - Get list of device types
- `getBrandTypes(brand)` - Get types for a specific brand
- `getModels(brand, type)` - Get models for brand/type combination
- `getFunctions(codesetId)` - Get available functions for a codeset
- `downloadCode(codesetId, functionName, apiKey)` - Download specific IR code
- `downloadCodeset(codesetId, apiKey)` - Download entire codeset

API Base URL: `https://irdb.globalcache.com:8081`

Database Logger (`src/lib/database-logger.ts`)

Provides centralized logging for all database operations with structured output.

Functions:
- `logDatabaseOperation(category, operation, data)` - Log database operations
- `logError(category, error, context)` - Log errors with context
- `logInfo(category, message, data)` - Log informational messages
- `logWarning(category, message, data)` - Log warnings

API Routes

IR Devices

`GET /api/ir/devices`

List all IR devices with their ports and commands.

Response:

```
{
  "success": true,
  "devices": [...]
}
```

`POST /api/ir/devices`

Create a new IR device.

Request Body:

```
{
  "name": "Cable Box 1",
  "deviceType": "Cable Box",
  "brand": "DirectTV",
  "model": "HR54",
  "matrixInput": 1,
  "matrixInputLabel": "Cable 1",
  "description": "Optional notes"
}
```

`GET /api/ir/devices/:id`

Get specific IR device with commands.

`PUT /api/ir/devices/:id`

Update an IR device.

`DELETE /api/ir/devices/:id`

Delete an IR device and all its commands.

IR Commands

`POST /api/ir/commands`

Create a new IR command.

Request Body:

```
{
  "deviceId": "...",
  "functionName": "POWER",
  "irCode": "sendir,1:1,1,38000,1,1,128,63,...",
  "category": "Power"
}
```

`DELETE /api/ir/commands?id=xxx`

Delete an IR command.

IR Database

`GET /api/ir/database/brands`

Get list of all brands from Global Cache database.

Response:

```
{
  "success": true,
  "brands": [
    { "Name": "Sony" },
    { "Name": "Samsung" }
  ]
}
```

`GET /api/ir/database/types?brand=xxx`

Get device types for a brand.

`GET /api/ir/database/models?brand=xxx&type=xxx`

Get models for brand and type.

Response:

```json
{
  "success": true,
  "models": [
    {
      "ID": "665",
      "Brand": "Sony",
      "Type": "DVD",
      "Name": "RDRHX Series",
      "Notes": ""
    }
  ]
}
```

`GET /api/ir/database/functions?codesetId=xxx`

Get available functions for a codeset.

Response:

```json
{
  "success": true,
  "functions": [
    { "SetID": "665", "Function": "POWER" },
    { "SetID": "665", "Function": "CHANNEL UP" }
  ]
}
```

`POST /api/ir/database/download`

Download IR codes for selected functions.

Request Body:

```json
{
  "deviceId": "...",
  "codesetId": "665",
  "functions": [
    { "functionName": "POWER", "category": "Power" },
    { "functionName": "CHANNEL UP", "category": "Channel" }
  ]
}
```

Response:

```json
{
  "success": true,
  "downloadedCount": 2,
  "commands": [...],
  "errors": []
}
```

Credentials

`GET /api/ir/credentials`

Get current credentials status.

Response:

```
{
  "success": true,
  "hasCredentials": true,
  "isLoggedIn": true,
  "email": "user@example.com"
}
```

`POST /api/ir/credentials`

Save and verify credentials.

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Frontend Components

Device Setup Page (`/device-setup`)

Main entry point for device configuration. Features:
- Tab navigation between Global Cache and IR devices
- Workflow instructions
- Links to Global Cache support and account creation

Global Cache Control Component

Manages Global Cache iTach devices:
- Add/edit/delete Global Cache devices
- Test connectivity
- Manage port assignments
- View device status (online/offline)

IR Device Setup Component

Manages IR-controlled devices:
- Add/edit/delete IR devices
- Configure device properties
- Link to matrix inputs
- Search IR database
- View command count
- Access to IR Database Search

IR Database Search Component

Three-step wizard for downloading IR codes:

Step 1: Credentials

- Login to Global Cache account
- Validates credentials
- Stores encrypted credentials

Step 2: Search

- Three-column interface:
  1. **Brands** - Search and select brand
  2. **Types** - Select device type for chosen brand
  3. **Models** - Select specific model
- Real-time filtering
- Progressive navigation

Step 3: Functions

- Grid display of available functions
- Category badges (Power, Volume, Channel, etc.)
- Select all/deselect all
- Batch download
- Progress indicator
- Error reporting

User Workflow

Complete Setup Process

1. **Setup Global Cache Device**
   - Navigate to Device Setup page
   - Go to "Global Cache Devices" tab
   - Click "Add Device"
   - Enter device details:

     - Name (e.g., "Global Cache 1")
     - IP Address (e.g., "192.168.5.110")
     - Port (default: 4998)
     - Model (optional)
     - Click "Add Device"
     - System tests connectivity
     - Device appears in list with status badge
     - Three IR ports automatically created

2. **Add IR-Controlled Device**
   - Go to "IR Controlled Devices" tab
   - Click "Add IR Device"
   - Enter device details:

     - Name (e.g., "Cable Box 1")
     - Device Type (from dropdown)
     - Brand (e.g., "DirectTV")
     - Model (optional)
     - Matrix Input (channel number)

   - Matrix Input Label (optional)
   - Description (optional)
   - Click “Add Device”

3. **Download IR Commands**
   - Click “IR Database” button on device card
   - **Step 1: Login**
     - Enter Global Cache account credentials
     - System validates and stores credentials
     - **Step 2: Search**
     - Select brand (pre-filled with device brand)
     - Select device type
     - Select model
     - Click model to proceed
     - **Step 3: Download**
     - Review available functions
     - Select desired commands (or Select All)
     - Click “Download”
     - Wait for download to complete
     - Review results (success/error counts)
     - System automatically:
     - Stores commands in database
     - Links to device
     - Updates codeset ID
4. **Link Device to Global Cache Port**
   - Return to “Global Cache Devices” tab
   - Find the Global Cache device
   - Locate the port to use
   - In the port’s assignment field, enter the IR device name
   - Port is now linked to the IR device
5. **Use Commands**
   - Commands are now available via the system’s IR control APIs
   - Can be sent using `/api/globalcache/send` endpoint

Logging System

All operations are logged with verbose output for debugging:

Log Format

```
[TIMESTAMP] [CATEGORY] operation_name
{
  "key": "value",
  "additionalData": "..."
}
```

Log Categories

- `IR_DEVICES` - IR device CRUD operations
- `IR_COMMANDS` - Command management
- `IR_DATABASE` - Cloud API operations
- `IR_DATABASE_API` - API route operations
- `IR_CREDENTIALS` - Credential management
- `GLOBAL_CACHE` - Global Cache operations

Example Log Output

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
📋 [IR DATABASE] Fetching models
   Brand: DirectTV
   Type: Satellite Receiver
   Timestamp: 2025-10-17T03:15:22.000Z
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
✅ [IR DATABASE] Models fetched successfully
   Count: 15
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

Security Considerations

1. **Password Encryption**
   - Passwords are encrypted using AES-256-CTR
   - Encryption key stored in environment variable `ENCRYPTION_KEY`
   - Should use proper key management in production

2. **API Key Storage**
   - Global Cache API keys are session-based
   - Stored temporarily in database
   - Cleared on logout

3. **Rate Limiting**
   - Daily download limit tracked per account
   - Prevents excessive API usage

Deployment

Prerequisites

- Node.js 18+
- npm or yarn
- PostgreSQL or SQLite database
- Global Cache account (for IR code downloads)

Installation Steps

1. **Database Migration**
   `bash`
   ```
     npx prisma generate
     npx prisma db push
   ```

2. **Build Application**

   ```
   bash
      npm run build
   ```

3. **Start Server**

   ```
   bash
      pm2 restart all
   ```

4. **Verify Deployment**
   - Navigate to `/device-setup`
   - Verify both tabs load correctly
   - Test adding devices

Troubleshooting

Common Issues

1. **“Module not found: @/components/globalcache/GlobalCacheControl”**
   - Ensure GlobalCacheControl.tsx is copied to remote server
   - Check file path matches import

2. **“No active IR database credentials”**
   - User needs to login via IR Database Search
   - Check credentials are saved in database

3. **IR Database API errors**
   - Verify Global Cache account is valid
   - Check network connectivity to irdb.globalcache.com
   - Verify daily download limit not exceeded

4. **Database schema errors**
   - Run `npx prisma db push` to sync schema
   - Check for migration conflicts

Future Enhancements

1. **Port Assignment UI**
   - Direct port-to-device linking in UI
   - Drag-and-drop interface
   - Visual port mapping

2. **Command Testing**
   - Test commands directly from UI
   - Preview command before sending
   - Command history

3. **Bulk Operations**
   - Import/export device configurations
   - Bulk command download
   - Template system for common devices

4. **Advanced Search**
   - Fuzzy search for models

- Recently used devices
- Favorite devices

5. **Command Organization**
   - Custom command categories
   - Command macros (sequences)
   - Conditional commands

API Reference

For detailed API documentation, see:
- Global Cache IR Database API: https://irdb.globalcache.com:8081/
- PDF Documentation: `API-GlobalIRDB_ver1.pdf`

Support

- Global Cache Support: https://www.globalcache.com/support/
- IR Database Account: https://irdb.globalcache.com

Changelog

Version 1.0.0 (2025-10-17)

- Initial implementation
- Added IRDevice, IRCommand, and IRDatabaseCredentials models
- Created IR Database service
- Implemented complete API routes
- Built comprehensive UI components
- Added verbose logging throughout
- Deployed to production server

Document Version: 1.0
Last Updated: October 17, 2025
Author: DeepAgent (Abacus.AI)