Content Rotation System Documentation

Overview

The content rotation system ensures a balanced mix of content types when creating schedules or filling template gaps. It cycles through different duration categories (ID, SPOTS, SHORT_FORM, LONG_FORM) in a configurable order to maintain variety throughout the broadcast day.

Table of Contents

How It Works

Basic Concept

The rotation system works like a carousel: 1. Start with the first category in the rotation order 2. Select content from that category 3. Move to the next category 4. Repeat until the schedule/gaps are filled

Example

With rotation order: `['id', 'spots', 'short_form', 'long_form']` - 1st content: ID - 2nd content: SPOTS
- 3rd content: SHORT_FORM - 4th content: LONG_FORM - 5th content: ID (starts over) - …and so on

Duration Categories

- **ID**: < 15 minutes (typically station IDs, promos)
- **SPOTS**: 15-30 minutes
- **SHORT_FORM**: 30-60 minutes
- **LONG_FORM**: > 60 minutes

Configuration

Setting Rotation Order via UI

1. **Navigate to Scheduling Settings**
   - Go to the scheduling panel
   - Click on "Configure Category Rotation"
2. **Modify Rotation Order**
   - Drag and drop categories to reorder
   - Add duplicate categories to increase their frequency
   - Remove categories you don't want in rotation

3. **Save Configuration**
    - Click “Save Configuration”
    - The system will confirm the save

Manual Configuration

If the UI configuration isn’t working, you can manually edit the config file:

1. **Locate config.json**

    ```
    cd backend
    cat config.json
    ```

2. **Add/Update rotation_order**

    ```
    {
      "scheduling": {
        "rotation_order": ["id", "spots", "short_form", "long_form"],
        "default_export_server": "target",
        "default_export_path": "/mnt/md127/Schedules/Contributors/Jay",
        "max_consecutive_errors": 100
      }
    }
    ```

3. **Restart the backend**

    ```
    python app.py
    ```

Default Rotation Orders

Frontend Default: `['id', 'spots', 'short_form', 'long_form']` **Backend Default**: `['id', 'short_form', 'long_form', 'spots']`

 Important: Make sure to save your configuration to avoid using mismatched defaults!

Content Type Rotation

You can also add specific content types (like PSA, MAF, BMP) to the rotation order. When using content types: - They should be added in lowercase (e.g., ‘maf’, ‘psa’, ‘bmp’) - Each content type has its own replay delay to prevent excessive repetition - Default replay delays for content types range from 2-8 hours depending on content

Default Replay Delays (hours): - Duration Categories: - `id`: 1 hour - `spots`: 2 hours - `short_form`: 4 hours - `long_form`: 8 hours - Content Types: - `psa`, `an`, `atld`: 2 hours - `bmp`, `im`, `lm`, `pkg`, `pmo`, `szl`, `spp`: 3 hours - `maf`, `imow`, `ia`: 4 hours - `mtg`: 8 hours (meetings need longer delays)

Common Issues and Solutions

Issue 1: Rotation Order Not Saving

Symptom: After setting the rotation order in UI, it reverts to default

Solution: 1. Check if `scheduling` section exists in config.json 2. Ensure the ConfigManager is loading all config sections: `python     # In config_manager.py, _merge_config should include:     else:         # Add new keys that don't exist in defaults default[key] = value`

3. Verify auto-save is enabled in UI settings

Issue 2: Backend Not Using Configured Order

Symptom: Backend logs show "No rotation_order in config, using default"

Solution: 1. Ensure config.json has the rotation_order: `bash     cat backend/config.json | jq '.scheduling.rotation_order'`

2. Force reload configuration in scheduler:

```
# This is already implemented in scheduler_postgres.py
self._config_loaded = False
self._load_config_if_needed()
```

Issue 3: Fill Gaps Not Following Rotation

Symptom: Content appears in wrong order when filling template gaps

Solution: 1. Check the logs for "Using rotation order:" message 2. Verify rotation advances only after content is scheduled: `python     # Only advance after successful scheduling self._advance_rotation()`

Technical Architecture

Components

1. **Frontend (script.js)**
   - `updateRotationOrder()`: Updates UI display
   - `saveRotationConfig()`: Saves order to scheduleConfig
   - `saveScheduleConfig()`: Sends to backend via API
2. **Backend API (app.py)**
   - `/api/config`: Receives and saves configuration
   - `update_scheduling_settings()`: Updates ConfigManager
   - `scheduler_postgres.update_rotation_order()`: Updates live scheduler
3. **Configuration Manager (config_manager.py)**
   - `load_config()`: Loads from config.json
   - `_merge_config()`: Merges loaded config with defaults
   - `get_scheduling_settings()`: Returns scheduling config
   - `update_scheduling_settings()`: Updates and saves config
4. **Scheduler (scheduler_postgres.py)**
   - `_load_config_if_needed()`: Loads rotation order from config
   - `_get_next_duration_category()`: Returns next category (doesn't advance)
   - `_advance_rotation()`: Advances to next category
   - `_reset_rotation()`: Resets to beginning

Data Flow

```
UI Configuration → Frontend JS → Backend API → ConfigManager → config.json
                                      ↓
                                 Scheduler ▯  (loads on demand)
```

Key Implementation Details

1. **Singleton Scheduler**: The scheduler uses a singleton pattern, so configuration must be explicitly reloaded
2. **Rotation Logic**:
   - Get category first
   - Try to find content
   - Only advance rotation AFTER content is scheduled
   - This prevents skipping categories with no content
3. **Configuration Persistence**:
   - Saved to `backend/config.json`
   - Loaded on scheduler initialization
   - Force-reloaded before each schedule creation

Testing Your Configuration

1. Verify Configuration is Saved

```
# Check if rotation_order is in config
cat backend/config.json | jq '.scheduling.rotation_order'
```

Expected output:

```
[
  "id",
  "spots",
  "short_form",
  "long_form"
]
```

2. Test with Fill Gaps

1. Create or load a template
2. Click “Fill Schedule Gaps”
3. Check the backend logs for:

   ```
   Using rotation order: ['id', 'spots', 'short_form', 'long_form']
   ```

3. Verify Rotation in Results

Check that content follows the configured pattern: - 1st item: ID category - 2nd item: SPOTS category - 3rd item: SHORT_FORM category - 4th item: LONG_FORM category

Troubleshooting

Enable Debug Logging

Add these log statements to track rotation:

```
# In scheduler_postgres.py
logger.info(f"Current rotation index: {self.rotation_index}")
logger.info(f"Selected category: {category}")
logger.info(f"Content found: {len(available_content)}")
```

Check Configuration Loading

```
# Test configuration loading
from config_manager import ConfigManager
cm = ConfigManager()
print('Scheduling config:', cm.get_scheduling_settings())
```

Force Configuration Reload

```
# In your schedule creation functions
scheduler._config_loaded = False
scheduler._load_config_if_needed()
```

Common Log Messages

Good:

```
Loaded rotation order from config: ['id', 'spots', 'short_form', 'long_form']
Using rotation order: ['id', 'spots', 'short_form', 'long_form']
```

Bad:

```
No rotation_order in config, using default: ['id', 'short_form', 'long_form', 'spots']
```

Best Practices

1. **Always Save Configuration**: After making changes in the UI, ensure you click “Save Configuration”
2. **Verify After Changes**: Check the logs to confirm your rotation order is being used
3. **Use Consistent Categories**: Stick to the standard categories unless you have specific requirements
4. **Monitor Rotation Balance**: Check that all categories are being used fairly

5. **Handle Empty Categories**: The system will skip categories with no available content

Advanced Configuration

Custom Rotation Patterns

You can create custom patterns by repeating categories:

```
{
  "rotation_order": ["id", "id", "spots", "short_form", "id", "long_form"]
}
```

This gives more weight to ID content.

Category-Specific Replay Delays

Configure how long before content can replay:

```
{
  "replay_delays": {
    "id": 6,        // 6 hours
    "spots": 12,    // 12 hours
    "short_form": 24,  // 24 hours
    "long_form": 48    // 48 hours
  }
}
```

Support

If you encounter issues not covered here:

1. Check the backend logs for error messages
2. Verify your config.json is valid JSON
3. Ensure the backend has write permissions for config.json
4. Try manually setting the rotation_order in config.json
5. Restart both frontend and backend services

Last updated: December 2024