Seedance-1.0 Model Parameters Summary

API Endpoint

- **URL**: `https://api.kie.ai/api/v1/jobs/createTask`
- **Method**: POST
- **Authentication**: Bearer token in Authorization header

Model Identifier

- **Model Name**: `bytedance/v1-pro-image-to-video`
- **Type**: Image-to-Video Generation

Available Parameters

Required Parameters

Parameter	Data Type	Description	Example
`model`	String	Model identifier	`"bytedance/v1-pro-image-to-video"`
`input.prompt`	String	Text description of desired video output	`"A golden retriever dashing through..."`
`input.image_url`	String (URL)	URL to the input image for video generation	`”https://i.ytimg.com/vi/_zEJq9-AzZc/hq720.jpg?sqp=-oay-mwEhCK4-FEIIDSFryq4qpAxMIARUAAAAAGAE-lAADIQj0AgKJD&rs=AOn4CLB1qvtYr-v1SEdeYYax9b-vA3_uc9qw

Optional Parameters

Parameter	Data Type	Options/Range	Default	Description
`input.resolution`	String	`"480p"` or `"720p"`	-	Output video resolution
`input.duration`	String/Number	Up to `"10"` (seconds)	`"5"`	Video duration in seconds
`input.camera_fixed`	Boolean	`true` / `false`	`false`	Controls camera movement
`input.seed`	Integer	`-1` (random) or specific seed	`-1`	Reproducibility seed
`input.enable_safety_checker`	Boolean	`true` / `false`	`true`	Content safety filtering
`callBackUrl`	String (URL)	Any valid webhook URL	-	Async callback endpoint

---

Motion Control Options

Camera Control

- **`camera_fixed`**:
- `false` (default): Allows dynamic camera movement
- `true`: Locks camera position for more stable motion
- **Usage for loops**: Setting to `true` can help with seamless loops by reducing camera drift

Prompt-Based Motion

- Motion is primarily controlled through detailed text prompts
- Prompt should describe:
- Subject motion (e.g., “dashing”, “walking”, “spinning”)
- Camera angle and position
- Speed/motion blur effects
- Environmental dynamics

Frame Control Capabilities

First/Last Frame Matching

⚠️ **Important**: The API documentation does **not explicitly mention** first/last frame matching parameters.

Workarounds for Seamless Loops:
1. **Use** `camera_fixed: true` to minimize camera drift
2. **Craft circular motion prompts** (e.g., "object rotating 360 degrees", "camera orbiting around subject")
3. **Select images with symmetric/repeatable content**
4. **Use shorter durations** (5s) for easier looping
5. **Post-processing**: Blend first/last frames externally if needed

Duration Control

- **Maximum**: 10 seconds
- **Recommended for loops**: 5 seconds
- Easier to create seamless transitions
- Lower chance of content drift
- Faster generation time

Constraints & Limits

Resolution Constraints

- Only two options: **480p** or **720p**
- Higher resolution (720p) means:
- Better quality
- Longer processing time
- Larger file size

Duration Constraints

- **Maximum**: 10 seconds per generation
- **String format**: Pass as string `"5"` or `"10"`

Input Image Requirements

- Must be accessible via public URL
- Format: Likely JPG, PNG, WebP (based on example)
- Recommended: High-quality, well-composed images

Safety Checker

- When enabled (`true`), content may be filtered
- May reject:
- Violence
- Explicit content
- Copyrighted material
- Other policy violations

Recommended Values for Seamless Loop Generation

Optimal Configuration

```
{
  "model": "bytedance/v1-pro-image-to-video",
  "input": {
    "prompt": "[Describe circular/repeatable motion]",
    "image_url": "[Your image URL]",
    "resolution": "720p",
    "duration": "5",
    "camera_fixed": true,
    "seed": -1,
    "enable_safety_checker": true
  }
}
```

Prompt Engineering for Loops

Good Loop Prompt Patterns:
- ✅ "Gentle pulsing motion, subtle breathing"
- ✅ "Slowly rotating 360 degrees, smooth continuous rotation"
- ✅ "Swaying back and forth, rhythmic motion"
- ✅ "Ambient environment, clouds drifting, repeating pattern"
- ✅ "Water flowing in circles, continuous loop"

Avoid for Loops:
- ❌ Linear motion (walking away, approaching)
- ❌ Dramatic camera movements (zooming, panning)
- ❌ Progressive actions (opening, closing, appearing, disappearing)
- ❌ Scene changes or transitions

Response Format

Success Response

```
{
  "code": 200,
  "message": "success",
  "data": {
    "taskId": "task_12345678"
  }
}
```

Usage Notes

1. API returns a `taskId` immediately
2. Video generation happens asynchronously
3. Use `callBackUrl` for webhook notifications
4. Or poll status using the `taskId`

Key Takeaways for Loop Generation App

1. **No native loop guarantee**: API doesn’t have explicit loop parameters
2. **Strategy**: Rely on `camera_fixed`, circular prompts, and post-processing
3. **Seed control**: Use `-1` for variety or fixed seed for reproducibility
4. **Duration sweet spot**: 5 seconds balances quality and loop feasibility
5. **Resolution choice**: 720p for quality, 480p for speed
6. **Image selection crucial**: Symmetric, centered subjects work best
7. **Callback recommended**: For better UX in web app (async processing)

Additional Considerations for Web App

User Experience

- Display estimated generation time (1-3 minutes typical)
- Show progress indicator during processing
- Preview input image before generation
- Allow prompt templates for common loop types

Technical Implementation

- Implement webhook handler for `callBackUrl`
- Store `taskId` for status tracking
- Handle API errors gracefully
- Implement retry logic for failed requests
- Consider video post-processing for true seamless loops (crossfade first/last frames)

Cost/Rate Limiting

- Check API documentation for rate limits
- Implement usage tracking
- Consider queue system for multiple requests
- Cache results to avoid regeneration