紙芝居DSL コマンドリファレンス

目次

紙芝居DSL コマンドリファレンス

対象DSL: `kamishibai=3.1`
対象読者: 台本作者、教材作成者、開発者

1. 記法の基本

1.1 コマンド行

```
キー=値
```

例:

```
asset=Beach1,backdrop
```

1.2 アクション行

```
action=対象:命令:値
action=対象:命令:値:追加値
```

例:

```
action=Urashima:say:こんにちは:2
action=stage:Beach1
```

1.3 シーン区切り

```
---
```

`---` でシーンを区切ります。

1.4 コメント

```
# ここはコメント
```

`#` で始まる行はコメントです。

1.5 区切り文字

文字	用途
`=`	行内で最初に現れるものが、コマンドのキーと値を分ける
`:`	アクションの要素を分ける
`,`	リストや座標を分ける
`---`	シーンを分ける
`#`	コメント行を表す

コマンド行では、最初の `=` だけがキーと値の区切りです。値に含まれる2個目以降の `=` はそのまま保持されるため、セリフ本文や `registerBranch` の条件式で使用できます。

半角 `:` と `,` の解釈はコマンドごとに異なります。たとえば `say` と `think` では半角 `:` が本文と秒数の区切りになるため、本文中のコロンには全角 `：` を使用します。

2. トップレベルコマンド

2.1 kamishibai

```
kamishibai=3.1
```

項目	内容
役割	台本バージョンを示す
必須	必須
推奨位置	ファイルの先頭
値	`3.1`

2.2 `asset`：画像・音声・テキストアセットの登録

書式

```
asset=<アセット名>,<リソース識別子>
```

`asset`コマンドは、画像、音声、またはテキストを紙芝居内で使用するためのアセットとして登録します。

- **アセット名** 紙芝居の台本内で画像や音声を参照するときに使用する名前です。
- **リソース識別子** アセットの取得元を表します。HTTPまたはHTTPSのURLだけでなく、TurboWarpプロジェクト内のコスチューム、背景、音、ランタイムテキストを指定できます。

`asset=`の後に現れる**最初のカンマ**が、アセット名とリソース識別子の区切りです。

外部URLから登録する

```
asset=<アセット名>,https://<画像または音声のURL>
```

例：

```
asset=forest,https://example.com/images/forest.png
asset=opening-bgm,https://example.com/sounds/opening.mp3
```

HTTPまたはHTTPSのURLで指定したファイルは、ネットワークから読み込まれます。

画像や音声を外部URLから読み込む場合は、配信元のWebサーバがCORSによるアクセスを許可している必要があります。また、URL先のファイルが移動または削除されると、アセットを読み込めなくなることがあります。

プロジェクト内アセットの短縮指定

```
asset=<アセット名>,costume
asset=<アセット名>,costume:<スプライト名>
asset=<アセット名>,backdrop
asset=<アセット名>,sound
asset=<アセット名>,text
```

例：

```
asset=Turtle,costume
asset=Urashima-walk-1,costume:Urashima
asset=Beach1,backdrop
asset=Ocean Wave,sound
asset=Narration,text
```

短縮指定では、アセット名をコスチューム名、背景名、音名、テキスト名として使います。`costume`だけの場合はアセット名と同名のスプライトを、`costume:スプライト名`では指定スプライトを参照します。

プロジェクト内アセットの明示指定

```
asset=<アセット名>,costume:<スプライト名>:<コスチューム名>
asset=<アセット名>,backdrop:<背景名>
asset=<アセット名>,sound:<スプライト名>:<音名>
asset=<アセット名>,sound:@stage:<音名>
asset=<アセット名>,text:<テキスト名>
```

例：

```
asset=hero-front,costume:人物:通常
asset=forest,backdrop:森
asset=narration-01,sound:ナレーター:場面1
asset=opening-bgm,sound:@stage:オープニング
asset=main-caption,text:Narration
```

使用可能なリソース識別子

リソースの種類	書式
外部画像・外部音声	`http://...`または`https://...`
スプライトのコスチューム	`costume:<スプライト名>:<コスチューム名>`
ステージの背景	`backdrop`または`backdrop:<背景名>`
スプライトの音	`sound:<スプライト名>:<音名>`
ステージの音	`sound`または`sound:@stage:<音名>`
テキスト	`text`または`text:<テキスト名>`

名前に使用できる文字

コロンはリソース識別子の区切りに使用するため、次の名前には使用できません。

- スプライト名
- コスチューム名
- 背景名
- 音名
- テキスト名

注意事項

- 指定したスプライト、コスチューム、背景、音が存在しない場合は、アセット登録エラーになります。
- コスチュームと背景は画像アセットとして扱われます。
- スプライトまたはステージの音は音声アセットとして扱われます。
- テキストはAsset Managerのランタイムテキストアセットとして扱われます。
- 画像アセットを音声再生に使用した場合や、音声アセットを画像表示に使用した場合はエラーになります。
- 同じアセット名を再度登録した場合は、新しい登録内容で置き換えられます。
- プロジェクト内のコスチューム、背景、音を使用する場合、それらのデータは`.sb3`ファイル内に保存されるため、外部の画像・音声サーバは必要ありません。

2.3 setLoadingBackdrop

```
setLoadingBackdrop=Loading用背景アセット名
```

アセット読込中にステージへ表示する背景アセットを１件指定します。

```
asset=loadingBackground,https://example.com/loading/background.png
setLoadingBackdrop=loadingBackground
```

項目	内容
役割	Loading用背景の最優先読込とステージ表示を設定する
必須	任意。省略時は組み込みの真っ黒な`LoadingBackdrop`を使う
値	`asset`で定義した画像アセット名１件
読込順	Loading用背景、Loading用画像、通常アセットの順に読み込む
進捗	Loading用背景を除外した`完了数 / 総数`を吹き出しに表示する

指定名の前後の空白は無視されます。指定した名前が`asset`で定義されていない場合は読込エラーになります。読み込み開始時は組み込みの真っ黒な背景を即座に表示し、指定背景の読み込み完了後に差し替えます。

2.4 setLoadingCostume

```
setLoadingCostume=Loading用アセット名1,Loading用アセット名2,...
```

アセット読込中に特別な組み込みスプライト`Loading`へ表示する画像アセットを指定します。

```
asset=loading1,https://example.com/loading/loading1.png
asset=loading2,https://example.com/loading/loading2.png
asset=loading3,https://example.com/loading/loading3.png
setLoadingCostume=loading1,loading2,loading3
```

項目	内容
役割	Loading用画像の優先読込と切替表示を設定する
必須	任意。省略時は組み込み`Loading`コスチュームを使う
値	`asset`で定義した画像アセット名のカンマ区切りリスト
読込順	Loading用背景に続けて、指定アセットを記述順のまま先に読み込み、その後に通常アセットを読み込む
進捗	Loading用アセットを除外した`完了数 / 総数`を吹き出しに表示する
アニメーション	通常アセットの１始まりの読込番号で指定画像を循環選択する

たとえば3画像を指定した場合、通常アセット1、2、3、4件目には、それぞれ`loading1`、`loading2`、`loading3`、`loading1`が表示されます。カンマ前後の空白は無視され、重複名は1件として扱われます。指定した名前が`asset`で定義されていない場合は読込エラーになります。

進捗の吹き出しは、Loading画像とは別の固定アンカーから表示されます。指定画像の大きさや非透明部分の外形は、吹き出し位置に影響しません。

2.5 setPoseRecognitionSound

```
setPoseRecognitionSound=ポーズ認識中の効果音アセット名,認識成立時の効果音アセット名
```

各ポーズの認識中に再生する音声アセットと、ポーズ条件の成立時に再生する音声アセットを指定します。

```
asset=Clock Ticking,https://example.com/sounds/clock-ticking.mp3
asset=Sewing Machine,https://example.com/sounds/sewing-machine.mp3
setPoseRecognitionSound=Clock Ticking,Sewing Machine
```

項目	内容
役割	ポーズ認識中の開始音と認識成立時の効果音を設定する
必須	任意。コマンド省略時または各値が空文字の場合は対応する音が無音
値1	`asset`で定義した認識中の音声アセット名
値2	`asset`で定義した認識成立時の音声アセット名。省略可
第1音の開始	各ポーズの認識開始時にAsset Manager経由で再生する
第1音の停止	認識成功、スペース／Right／Downによるスキップ、物語停止時に停止する
第2音の開始	ポーズ条件成立時、「ポーズ認識」の値を更新する直前に再生する

指定した名前が音声アセットとして定義されていない場合は、再生時にアセットエラーになります。第1音の長いクリップは認識が先に終わると途中で停止します。第2音は音声アセット自身の長さだけ再生されます。従来の1音指定はそのまま利用できます。

2.6 actor

```
actor=アクター名,初期スキン名
```

項目	内容
役割	登場人物を登録する
必須	登場人物を表示する場合は必要
値1	アクター名
値2	初期スキン名

例:

```
actor=Urashima,Urashima-walk-1
actor=Turtle,Turtle
```

2.7 cover

```
cover=背景アセット名,音声アセット名
```

項目	内容
役割	表紙画面の背景と音を指定する
必須	推奨
値1	背景アセット名
値2	音声アセット名

例:

```
cover=Beach1,OceanWave
```

2.8 setRuntimeVariable

```
setRuntimeVariable=変数名:値
```

Temporary Variables拡張のランタイム変数へ値を設定します。ヘッダまたはシーン内で使用できます。

```
setRuntimeVariable=startSceneIndex:1
setRuntimeVariable=takeSeaRoute:true
```

`startSceneIndex` は最初に実行するシーン番号です。条件分岐で参照する変数もこのコマンドで初期化できます。

2.9 registerBranch

```
registerBranch=分岐名:条件1,条件2,...:シーンラベル1,シーンラベル2,...
```

Runtime Expression で評価する条件と、移動先シーンラベルの組を登録します。条件は左から評価され、最初に真になった条件と同じ位置のラベルが選ばれます。

```
registerBranch=chooseRoute:takeSeaRoute,true:ocean,home
```

等価比較を含む条件も指定できます。

```
setRuntimeVariable=score:1
setRuntimeVariable=route:ocean
registerBranch=chooseByScore:score == 1,true:ocean,home
registerBranch=chooseByName:route === "ocean",true:ocean,home
```

等価比較演算子は `==`、`!=`、`===`、`!==` です。単独の `=` は代入記号であり、条件式の演算子としては使用できません。条件数とラベル数はそろえてください。最後の条件に `true` を置くと既定の移動先になります。登録した分岐は `action=branch:分岐名` で実行します。

2.10 sceneLabel

```
sceneLabel=シーンラベル
```

`---` で区切られたシーンへ一意の名前を付けます。`branch`、`keyInputToChangeScene`、`touchInputToChangeScene` はこのラベルを移動先として使います。

```
---
sceneLabel=ocean
action=stage:Ocean
```

2.11 TMPoseURL

```
TMPoseURL=ポーズモデルURL
```

項目	内容
役割	TMPose モデルを読み込む
必須	`pose` を使うシーンでは原則必須
値	モデルURL
実行タイミング	シーン内のアクション実行前にモデル読み込みを行う

例:

```
TMPoseURL=https://example.com/kamishibai/pose-model/
```

2.12 text：scene 0 のポーズ案内

```
text=予約済みUIテキストアセット名:文字列
```

ポーズ案内は、scene 0（最初の `---` より前）で定義します。`ui.prompt` はランタイムが自動登録するため、`asset=` による登録は不要です。

```
text=ui.prompt:ポーズをとろう！
```

アセット名	用途	未定義時の既定値
`ui.prompt`	ポーズ認識中の案内	`Pose!`

有効な台本を開始するたびに、既定値へ戻した後でscene 0の定義を適用します。

ファイル読込、再読込、タイトル表示、言語選択、台本エラーの文言は台本コマンドではありません。アプリの言語定義と標準の `（言語）` ブロック、または利用者が保存した言語選択から決まります。旧台本の `text=ui.open:`、`text=ui.reload:`、`text=ui.about:`、`text=ui.invalidScript:` はアプリUIへ適用されません。

3. グローバルアクション

グローバルアクションは、特定のアクターではなく、ステージ、音、時間を操作します。

3.1 stage

```
action=stage:背景アセット名
```

背景を指定アセットに切り替えます。

例:

```
action=stage:Beach1
```

3.2 wait

```
action=wait:秒数
```

指定秒数だけ待ちます。

例:

```
action=wait:1.5
```

3.3 bgm

```
action=bgm:音声アセット名
```

音声アセットを再生し、再生完了を待たずに次のアクションへ進みます。

例:

```
action=bgm:Odesong
```

補足: `bgm` はループ再生しません。再生開始後すぐに次のアクションへ進み、次のシーンに移っても音声自体の再生が終わるまで継続します。

3.4 sound

```
action=sound:音声アセット名
```

音声アセットを再生し、再生完了まで待ちます。

例:

```
action=sound:Gong
```

効果音の演出を確実に聞かせたい場面や、音が終わってから次の場面へ進めたい場合に使います。

3.5 text

```
action=text:テキストアセット名:文字列
```

`asset=名前,text` で登録したテキストアセットの内容を、アクション列のその位置で更新します。`wait` と組み合わせると、内容を時系列に沿って順次変更できます。空文字列で内容を消せます。

```
action=text:Narration:むかし
action=wait:2
action=text:Narration:むかし　むかし、あるところに...
action=wait:2
action=text:Narration:
```

シーン直下の `text=...` も互換性のため読み込めますが、アクション列より先に処理されます。新しい台本や順次更新には `action=text:...` を使用してください。予約済みの `ui.*` 文言だけは初期設定としてscene 0の `text=...` を使用します。

3.6 transition

```
action=transition:fadeOut
action=transition:fadeUp
action=transition:reset
action=transition:fadeToWhite
action=transition:fadeFromWhite
```

ステージの明るさ効果を使って場面転換します。

値	動作
`fadeOut`	明るさを段階的に下げる
`fadeUp`	明るさを段階的に上げる
`reset`	明るさ効果を `0` へ戻す
`fadeToWhite`	明るさを `+100` まで段階的に上げ、その状態を保持する
`fadeFromWhite`	明るさを `+100` から `0` まで段階的に下げる

3.7 branch

```
action=branch:分岐名
```

`registerBranch` で登録した条件を評価し、選ばれた `sceneLabel` へ移動します。真になる条件がなければ、そのまま次のアクション／シーンへ進みます。

3.8 keyInputToChangeScene

```
action=keyInputToChangeScene:キーID1,キーID2,...:シーンラベル1,シーンラベル2,...
```

物理キーの入力をバックグラウンドで待ち、押されたキーと同じ位置のシーンラベルへ移動します。

```
action=keyInputToChangeScene:ArrowLeft,ArrowRight:leftRoute,rightRoute
```

キーIDは `KeyA`、`Space`、`ArrowLeft` などのコードを使います。キーID数とラベル数はそろえてください。

3.9 touchInputToChangeScene

```
action=touchInputToChangeScene:アクター名1,アクター名2,...:シーンラベル1,シーンラベル2,...
```

指定アクターへのポインター／タッチ入力を待ち、選ばれたアクターと同じ位置のシーンラベルへ移動します。アクター数とラベル数はそろえてください。

4. アクターアクション

アクターアクションは、`actor=` で登録した登場人物に対して実行します。

```
action=アクター名:命令:値
```

4.1 対象指定

指定	意味	例
単独アクター	1 人だけに実行	`Urashima`
カンマ区切り	複数アクターに実行	`Urashima,Turtle`
`*`	全アクターに実行	`*`

例:

```
action=Urashima,Turtle:hide
action=*:hide
```

複数指定や `*` は実装上サポートされていますが、台本の読みやすさを優先するなら、通常は単独指定をおすすめします。

4.2 show

```
action=アクター名:show:スキン名:x,y,サイズ
action=アクター名:show:x,y,サイズ
```

アクターのスキン、位置、サイズを指定して表示します。

例:

```
action=Urashima:show:Urashima-walk-1:172,-77,25
```

値	意味
スキン名	`asset` で登録した画像アセット名
x	横位置。右がプラス、左がマイナス
y	縦位置。上がプラス、下がマイナス
サイズ	Scratch/TurboWarpのスプライトサイズ

スキン名を省略した書式では、`actor` で指定した初期スキンまたは現在のスキンを使います。

4.3 hide

```
action=アクター名:hide
```

アクターを非表示にします。

例:

```
action=Princess:hide
```

4.4 say

```
action=アクター名:say:セリフ
action=アクター名:say:セリフ:秒数
```

セリフ吹き出しを表示します。

例:

```
action=Princess:say:ようこそ竜宮城へ。:2.5
```

吹き出しを消す例:

```
action=Urashima:say:
```

4.5 think

```
action=アクター名:think:文章
action=アクター名:think:文章:秒数
```

思考吹き出しを表示します。

例:

```
action=Urashima:think:あっという間におじいさんになってしまった…:3
```

4.6 setSkin

```
action=アクター名:setSkin:スキン名
action=アクター名:setSkin:スキン名:サイズ
```

アクターのスキンを、登録済みアセットに切り替えます。

例:

```
action=Urashima:setSkin:Urashima-surprised
action=Urashima:setSkin:Urashima-surprised:45
```

4.7 setScale

```
action=アクター名:setScale:サイズ
```

アクターのサイズを変更します。

例:

```
action=Urashima:setScale:30
```

4.8 setPosition

```
action=アクター名:setPosition:x,y
```

アクターを指定位置へ瞬時に移動します。

例:

```
action=Urashima:setPosition:0,-57
```

4.9 moveTo

```
action=アクター名:moveTo:x,y,秒数
```

指定秒数をかけて、アクターを指定位置へ移動します。

例:

```
action=Urashima:moveTo:40,-57,1.5
```

4.10 setLayer

```
action=アクター名:setLayer:front
action=アクター名:setLayer:back
action=アクター名:setLayer:数値
```

アクターの重なり順を変更します。

値	動作
`front`	最前面へ移動
`back`	最背面へ移動
正の数値	指定レイヤー数だけ前へ移動
`0`、負の数値、数値以外	1レイヤー後ろへ移動

例:

```
action=Princess:setLayer:front
action=Turtle:setLayer:back
```

4.11 loop

```
action=アクター名:loop:アセット1,アセット2,...:秒数1,秒数2,...
```

画像または音声アセットをバックグラウンドで繰り返します。秒数は各アセットの開始から次のアセット開始までの間隔で、アセット数と同じ個数を指定します。最後の秒数の後に先頭へ戻ります。

```
action=Fish:loop:Fish1,Fish2:1,1
```

秒数 `0` で複数アセットを同時に開始できます。同時グループに複数の画像がある場合は最後の画像が表示されます。

4.12 sequence

```
action=アクター名:sequence:アセット1,アセット2,...:秒数1,秒数2,...
```

画像または音声アセットをバックグラウンドで一回だけ順番に再生します。秒数はアセット数より１つ少なくします。コマンド自体は完了を待たず、次のアクションへ進みます。

```
action=Hero:sequence:Hero1,StepSound,Hero2:0,0.5
```

5. ポーズ認識アクション

5.1 基本形

```
action=アクター名:pose:スキン名リスト:ポーズ名リスト:効果音リスト
```

例:

```
action=Urashima:pose:Urashima-help-1:help:SquishPop
```

5.2 複数ポーズ

```
action=Urashima:pose:Skin1,Skin2,Skin3:pose1,pose2,pose3:Sound1,Sound2,Sound3
```

処理回数はポーズ名の個数で決まります。各回では、同じ位置のスキンと効果音を参照します。対応する項目がない場合は空文字列になり、ポーズ名より後ろにある余分なスキンや効果音は参照されません。そのため、各ポーズに対応するスキンと効果音を１つずつ指定します。

項目	説明
スキン名リスト	各ポーズ待ちのときに表示するアクター画像
ポーズ名リスト	TMPoseモデルに登録されたラベル名
効果音リスト	各ポーズ成功時に鳴らす音

5.3 実行時の流れ

1. カメラプレビューを表示します。
2. ポーズ認識を開始します。
3. プロンプトと変数 `ポーズ認識`、`チャージ` を表示します。
4. 対象アクターのスキンを、現在のポーズ用スキンに変更します。
5. 指定ポーズが認識されるまで待ちます。
6. 認識されている間、`チャージ` が増えます。
7. `チャージ` が100になると成功です。
8. 成功音を鳴らします。
9. 次のポーズがあれば繰り返します。
10. すべて完了したら、ポーズ認識とカメラプレビューを閉じます。

5.4 認識パラメータの実装値

現在の配布用アプリは、起動時に `poseRecog=0.5`、`poseCharge=10`、`poseIdle=0` を設定し、次の計算で進行します。

項目	内容
認識しきい値	対象ポーズのスコアが `0.5` 以上なら認識中と判定する
認識中	反復ごとに `min(対象ポーズのスコア × 10, 100)` をチャージへ加える
非認識中	`poseIdle=0` のためチャージを増やさない
成功条件	チャージが100に達すると次のポーズへ進む

しきい値未満ではチャージは増えません。認識中は対象ポーズのスコアに応じてチャージが増えるため、ポーズを保つと成功条件へ到達します。

5.5 ポーズ設計のコツ

よいポーズ	避けたいポーズ
腕や体の位置がはっきり違う	似た姿勢が多い
正面から分かりやすい	横向きで見えにくい
子供でもまねしやすい	難しすぎる、危ない
1〜2秒止まれる	素早すぎる動き

6. 座標とサイズ

TurboWarpのステージ座標は、中央が `(0, 0)` です。

方向	値
右	xがプラス
左	xがマイナス
上	yがプラス
下	yがマイナス

例:

```
action=Urashima:show:Urashima-walk-1:0,-60,30
```

これは、浦島を中央やや下に、サイズ30で表示します。

7. シーン終了時の挙動

各シーン終了時に、アプリは次の処理を行います。

- アクションリストをクリアする
- すべてのアクターを隠す
- 背景と画面効果を現在値のまま保持する
- `bgm` で開始した再生を止めず、音声自体の終了まで継続する

したがって、次のシーンでは背景とアクター表示を必要に応じて書き直します。同じBGMを継続する場合は、`bgm`を指定し直しません。

通常は次のシーン番号へ進みます。`branch` またはキー／タッチ入力が `nextSceneLabel` を設定した場合は、`sceneLabelList`から同名ラベルを探し、そのシーンへ移動します。

8. 台本エラーになりやすい例

8.1 未対応コマンド

```
background=Beach1
```

`background` は未対応です。正しくは次のように書きます。

```
action=stage:Beach1
```

8.2 `=` がない

```
asset Beach1,https://example.com/Beach1.png
```

正しくは次のように書きます。

```
asset=Beach1,https://example.com/Beach1.png
```

8.3 `:` が不足している

```
action=Urashima say こんにちは
```

正しくは次のように書きます。

```
action=Urashima:say:こんにちは:2
```

8.4 未登録アセットを参照する

```
action=stage:Beach99
```

`Beach99` を使うなら、ヘッダで定義しておく必要があります。

```
asset=Beach99,https://example.com/Beach99.png
```

8.5 存在しないシーンラベルを参照する

```
action=keyInputToChangeScene:ArrowRight:missingScene
```

移動先には、いずれかのシーンで定義した `sceneLabel` を指定します。ラベルは大文字・小文字や空白も含めて一致させてください。

8.6 対応リストの個数が違う

```
registerBranch=route:routeA,routeB:sceneA
```

条件とラベル、キーIDとラベル、タッチ対象とラベル、`loop`のアセットと秒数は、それぞれ必要な個数をそろえます。

9. リハーサル用キー

キー	実装上の効果	使いどころ
スペース	現在のポーズ認識1件をスキップ	複数ポーズを1件ずつ確認する
右矢印	現在のアクションを完了して次のアクションへ進む	演出を1件ずつ確認する
下矢印	現在シーンを終了して次のシーンへ進む	長いシーンを飛ばす

スペースはポーズ待ち中、右矢印はアクション実行中、下矢印はシーン実行中だけ受け付けます。未処理の入力要求がある間は後続キーで上書きしません。タイトル画面ではスペースまたは画面クリックだけが有効です。

下矢印では、現在実行中とシーン残部の`transition`を待ち時間なしで最終状態まで適用し、シーン残部の`bgm`を再生開始します。再生中のBGMは維持し、現在実行中の`sound`だけを停止します。残りのその他のアクションは実行しません。

本番運用では誤操作に注意してください。

10. チートシート

10.1 ヘッダ

```
kamishibai=3.1
setRuntimeVariable=startSceneIndex:1
asset=Backdrop,backdrop
asset=ActorSkin,costume:Actor
asset=Music,sound
asset=Narration,text
asset=Loading1,https://example.com/loading1.png
asset=Loading2,https://example.com/loading2.png
asset=LoadingBackground,https://example.com/loading-background.png
asset=ClockTicking,https://example.com/clock-ticking.mp3
asset=PoseRecognized,https://example.com/pose-recognized.mp3
setLoadingBackdrop=LoadingBackground
setLoadingCostume=Loading1,Loading2
setPoseRecognitionSound=ClockTicking,PoseRecognized
text=ui.prompt:ポーズをとろう！
actor=ActorName,InitialSkin
cover=CoverBackground,CoverSound
registerBranch=route:flag,true:sceneA,sceneB
```

10.2 シーン

```
---
# scene 1
sceneLabel=opening
TMPoseURL=https://example.com/model/
action=stage:Background
action=transition:fadeUp
action=transition:fadeToWhite
action=stage:NextBackground
action=transition:fadeFromWhite
action=wait:1
action=Actor:show:Skin:x,y,scale
action=Actor:say:Text:seconds
action=Actor:moveTo:x,y,seconds
action=Actor:loop:Skin1,Skin2:0.5,0.5
action=Actor:pose:Skin1,Skin2:pose1,pose2:Sound1,Sound2
action=branch:route
```

10.3 よく使うアクション

```
action=stage:Beach1
action=wait:1.5
action=bgm:Music
action=sound:Effect
action=text:Narration:場面の説明
action=Hero:show:Hero-normal:0,-60,30
action=Hero:say:こんにちは:2
action=Hero:think:どうしよう:2
action=Hero:setSkin:Hero-happy
action=Hero:moveTo:100,-60,1
action=Hero:setPosition:0,-60
action=Hero:hide
action=Hero:sequence:Hero1,StepSound,Hero2:0,0.5
action=Hero:pose:Hero-help:help:Success
action=keyInputToChangeScene:ArrowLeft,ArrowRight:left,right
```

11. 推奨命名規則

対象	例
背景	`Beach1`, `Ocean`, `DragonCastle`, `End`
キャラクター通常	`Urashima-walk-1`
キャラクター動作	`Urashima-open-1`, `Urashima-open-2`
感情	`Urashima-surprised`, `Hero-happy`
音	`OceanWave`, `GoalCheer`, `Jump`
ポーズ	`help`, `ride1`, `dance2`, `open3`, `despair`

12. 互換性メモ

このリファレンスは、提供されたTurboWarpプロジェクトの `kamishibai=3.1` 実装を前提にしています。3.1より前の台本では、アセット識別子、シーンラベル、分岐、入力、トランジション、アニメーション、テキストが異なる、または利用できない場合があります。台本を配布する場合は、台本ファイルと対応するアプリのバージョンを一緒に管理してください。

13. 関連ドキュメント

- `03-user-guide.md`: 紙芝居アプリの操作方法
- `04-dsl-manual.md`: 紙芝居DSLファイルの作り方
- `01-executive-summary-adult.md`: 大人向け概要説明
- `02-executive-summary-kids.md`: 子供向け概要説明
- `06-developer-guide.md`: 成果物とビルダーの利用、開発、検証、公開の手順
- `07-internal-specification.md`: 汎用アプリSB3の内部構造、呼出し関係、状態遷移
- `history.md`: 紙芝居DSL 2.0から3.1への変更履歴