紙芝居アプリ ソフトウェアメンテナンスガイド

目次

紙芝居アプリ ソフトウェアメンテナンスガイド

Copyright © 2026 Hiroya Kubo. この文書はCC BY-SA 4.0で提供します。

このガイドは、TMPose紙芝居の成果物とビルダーを利用し、アプリ、SB3ソース、 ビルダー、Webサイト、ドキュメントを変更・検証・公開するソフトウェア開発者向けの 作業資料です。次を本書の責務とします。

- 成果物プロファイルとSB3・台本変換ビルダーの外部契約
- リポジトリの開発環境、構成、共通フロー、変更対象別手順
- テスト、成果物検証、GitHub Pagesとnpmの公開手順
- 開発時のトラブルシューティング、ライセンス、秘密情報の扱い

汎用アプリSB3のtarget、変数、event、custom block、呼出し関係、状態遷移は 紙芝居アプリ内部仕様書を正本とします。台本の書式と コマンド仕様は台本DSLマニュアルと コマンドリファレンスを参照してください。本書には、 これらの内部構造やDSL項目を重複して列挙しません。

対象アプリ／DSL: `kamishibai=3.1`

過去のバージョンからの変更は`history.md`を参照してください。

このガイドの章は、次の５つの区分で並んでいます。

区分	対象	読み方
導入	管理範囲、開発環境、リポジトリ構成、共通フロー	初めて開発するときに、この順に読む
利用契約	成果物プロファイル、ビルダーのCLI／API／manifest	成果物を生成・利用するときに参照する
変更対象別手順	アプリSB3、機能拡張、ビルダー実装、文書とサイト	変更対象に応じて、必要な章だけを読む
検証と公開	自動・手動検証、GitHub Pages、npm、障害時の扱い	PRとリリースの完了条件として読む
参照	関連プロジェクト、ライセンス、秘密情報、関連ドキュメント	関連する規約や資料を探すときに参照する

「導入」では、管理範囲を確認し、開発環境とリポジトリを把握してから、共通の 開発フローへ進みます。「利用契約」はビルダー実装の変更手順ではありません。 「変更対象別手順」は、この順に実施する一連の工程ではなく、変更対象ごとに独立した 手順です。「検証と公開」は対象変更に必要な確認を選び、標準チェックへ合流します。

1. 管理範囲と責務を理解する

このリポジトリが管理するものは次のとおりです。

- 物語固有の台本やアセットを含まない、汎用の紙芝居アプリ
- アプリ SB3 の展開ソースと、配布用 SB3 を生成する設定
- 台本と画像・音声を SB3 へ組み込む npm パッケージ
- GitHub Pages へ公開する Web サイトと一般向けドキュメント
- 上記を検証する自動テスト

関連プロジェクトとの境界は次のとおりです。

対象	管理場所	このリポジトリとの関係
SB3 の展開・検証・決定的再構築	`kubohiroya/sb3-toolchain`	固定依存として利用する。「sb3-toolchain」（15.1 節）を参照
浦島太郎などの公開用物語	`kubohiroya/tmpose-kamishibai-samples`	`stories/urashima/` で台本、固有アセット、生成物を管理する
埋め込み機能拡張	各機能拡張の GitHub リポジトリ	`app/` には検証済み成果物と由来情報だけを同期する
TurboWarp Extension Gallery の機能拡張	Gallery の公開 URL	SB3 から外部 URL を参照する

公開サンプル の固有ファイルを本体へコピーしません。本体の汎用性と、サンプルの独立した更新・配布を維持します。

依存バージョン、スクリプト、公開対象の正本は `package.json` と `pnpm-lock.yaml` です。 文書には特定 commit を転記せず、必要なときに確認します。

```
pnpm why @kubohiroya/sb3-toolchain
git diff -- package.json pnpm-lock.yaml
```

2. 開発環境を準備する

2.1 必要な環境

- Node.js 22.12.0以上
- pnpm 11
- PDF生成に利用できるChromeまたはChromium
- Gitと、GitHub操作に利用するGitHub CLI

```
corepack enable
pnpm install
```

CIではlockfile以外の依存解決を許可しません。

```
pnpm install --frozen-lockfile
```

初回セットアップ後は、展開SB3ソースとテスト用SB3を確認します。

```
pnpm sb3:check
pnpm test
```

macOSでは通常のGoogle ChromeをPDF生成に自動利用します。別のブラウザを使う環境では、`VIVLIOSTYLE_CHROME_PATH`へ実行ファイルの絶対パスを設定します。

3. リポジトリ構成を把握する

パス	役割
`app/`	紙芝居アプリ SB3 の Git 管理上の正本
`app/project.source.json`	整形済み Scratch プロジェクト
`app/assets/`	汎用アプリ自身が使用する画像・音声
`app/extensions/`	埋め込み機能拡張の同期済み JavaScript
`app/embedded-extensions.json`	埋め込み機能拡張の管理情報
`app/sb3-source.json`	SB3 展開ソースのマニフェスト
`src/builder/`	npm で公開する SB3・台本変換 API
`bin/`	npm CLI のエントリーポイント
`scripts/`	サイト、ドキュメント、配布物のビルドと検証
`docs/general/`	一般向け・開発者向けの文書原稿
`docs/workshops/`	日付付き体験会資料
`site/`	GitHub Pages の静的入力
`test/`	ビルダー、SB3、VM、ドキュメント、公開契約のテスト

次の場所は生成物であり、Git 管理上の正本ではありません。

パス	内容
`tmp/kamishibai.sb3`	TurboWarp 編集と自動テストに使う SB3
`dist/`	GitHub Pages へ公開するサイト、HTML/PDF、配布用 SB3
`output/pdf/`	印刷用 PDF のローカル確認先

4. 共通の開発フローに従う

すべての変更は GitHub Issue へ受け入れ基準とロールバック手順を記録し、小さなブランチと PR へ分けます。

1. 最新の`main`から作業ブランチを作る。
2. `pnpm install --frozen-lockfile`で依存を復元する。
3. 変更対象に近いテストを先に実行する。
4. 生成物ではなく正本を変更する。
5. 差分と生成結果を確認する。
6. 標準チェックと必要な手動確認を行う。
7. Issueの運用ログとDoDを更新してPRを作成する。

無関係な変更や未追跡ファイルをまとめてコミットしません。SB3のimportや成果物の 置換を行う前には、必ず`git status`と対象パスの差分を確認します。変更対象ごとの テストと公開前チェックは「変更を検証する」（第11章）を 参照してください。

5. 成果物プロファイルを理解する

紙芝居の成果物は、台本と物語固有アセットをどこに保持するかで分けます。

プロファイル	例	台本	物語固有アセット	主な用途
`generic`	`kamishibai.sb3`	非埋め込み	非埋め込み	本体が配布する汎用雛形
`editor`	`_urashima.sb3`	非埋め込み	埋め込み	物語作成者の編集・動作確認
`player`	`urashima.sb3`	埋め込み	埋め込み	配布・再生、Packager Web版

`generic`は`app/`から生成し、特定の物語を含めません。builder APIとCLIが受け付ける `profile`は`editor`または`player`です。

`editor`と`player`は同じベースSB3、台本、アセットロックから生成します。両者の 変換済み台本とアセット参照を分岐させません。`player`は組み込み台本を予約変数へ保存し、 タイトル操作後にファイル選択なしで開始します。

`player`へ台本とアセットを組み込んでも、TMPoseモデル、カメラ、外部サービスまで 自動的にオフライン化されるわけではありません。残るオンライン依存は成果物manifestと 公開ページへ明記します。

6. SB3・台本変換ビルダーを利用する

この章はビルダー利用者に対する外部契約を示します。ビルダーの実装を変更する手順は 「ビルダーの実装を変更する」（第9章）を 参照してください。汎用アプリSB3の内部構造は本章の対象外です。

6.1 導入

利用可能なバージョンを確認し、消費側で明示的に固定します。

```
npm view @kubohiroya/tmpose-kamishibai version
pnpm add --save-exact @kubohiroya/tmpose-kamishibai@<VERSION>
```

生成したlockfileをcommitし、CIでは`pnpm install --frozen-lockfile`を使います。

6.2 CLI

```
pnpm exec tmpose-kamishibai build-sb3 \
  --base kamishibai.sb3 \
  --script source.txt \
  --assets assets.lock.json \
  --output dist/sample \
  --profile editor
```

`--output`は拡張子を含まないベース名です。次の３ファイルを同じtransactionとして 生成します。

```
dist/sample.sb3
dist/sample.txt
dist/sample.manifest.json
```

オプション	意味
`--allow-file-root DIR`	`file:`の許可ルートを追加。複数回指定可能
`--allow-http`	平文HTTPを明示的に許可
`--timeout-ms N`	1リクエストのタイムアウト
`--max-asset-bytes N`	1アセットの最大バイト数
`--max-script-bytes N`	組み込み台本の最大バイト数
`--max-redirects N`	HTTPリダイレクト上限

完全な一覧は`pnpm exec tmpose-kamishibai --help`で確認します。

6.3 JavaScript API

```
import {
  Sb3BuilderError,
  buildSb3Bundle,
  validateAssetManifest,
  validateBundle,
} from '@kubohiroya/tmpose-kamishibai/builder';

const result = await buildSb3Bundle({
  baseSb3: 'kamishibai.sb3',
  sourceScript: 'source.txt',
  assetManifest: 'assets.lock.json',
  outputDirectory: 'dist',
  outputName: 'sample',
  profile: 'editor',
});

console.log(result.outputPaths);
```

`baseSb3` と `sourceScript` にはファイルパスまたは `file:` URL を指定できます。 `assetManifest` にはファイルパス、 `file:` URL、または検証対象の JavaScript オブジェクトを 指定できます。相対 `file:` を含むオブジェクトでは `manifestBaseDirectory` も指定します。

ネットワーク・ファイル取得は `allowedFileRoots` 、 `allowHttp` 、 `requestTimeoutMs` 、 `maxAssetBytes` 、 `maxRedirects` で制限できます。 `player` の組み込み台本上限は `maxEmbeddedScriptBytes` で変更できます。

`buildSb3Bundle` は `manifest` と `outputPaths` を返します。入力・アセット・出力の問題は `Sb3BuilderError` として処理段階とアセット情報を保持します。

6.4 アセットマニフェスト

入力 manifest は `formatVersion: 1` と 1 件以上の `assets` を持ちます。

```
{
  "formatVersion": 1,
  "assets": [
    {
      "name": "forest",
      "uri": "file:assets/forest.svg",
      "kind": "backdrop",
      "target": "@stage",
      "sb3Name": "森",
      "contentType": "image/svg+xml",
      "dataFormat": "svg",
      "size": 1234,
      "sha256": "<64文字の16進数>",
      "license": "CC-BY-4.0: https://creativecommons.org/licenses/by/4.0/",
      "metadata": {
        "bitmapResolution": 1,
        "rotationCenterX": 240,
        "rotationCenterY": 180
      }
    }
  ]
}
```

kind	target	変換後の台本参照
`backdrop`	`@stage`	`backdrop:<sb3Name>`
`costume`	スプライト名	`costume:<target>:<sb3Name>`
`stageSound`	`@stage`	`sound:@stage:<sb3Name>`
`spriteSound`	スプライト名	`sound:<target>:<sb3Name>`

DSL名、同一target内のSB3名、既存SB3のアセット名は重複できません。`license`には素材の ライセンスまたは利用条件の識別情報と参照先を記録します。

6.5 安全性と再現性

- `file:`は既定でmanifestのディレクトリ以下だけを許可し、`..`やsymlinkによる脱出を拒否する
- HTTPSを既定とし、平文HTTPは明示的に許可した場合だけ取得する
- Content-Type、実サイズ、ロック済みサイズ、SHA-256、timeout、redirectを検証する
- ZIP entry順、timestamp、圧縮設定、JSON表現を固定する

- SB3、変換済み台本、出力manifestの対応を確定前に再検証する
- 3成果物を一時領域で生成し、すべて成功した場合だけ置換する
- 失敗時は既存成果物を保持または復元する

同じ入力、固定依存、設定から生成したSB3、台本、manifestはbit-for-bitで一致しなければ なりません。

7. アプリSB3を変更する

このリポジトリでは`app/`をアプリSB3の正本とし、固定した`sb3-toolchain`を`pnpm sb3:*`スクリプトから利用します。展開ソース形式、importとbuildの上書き保護、 決定的出力の共通仕様は`sb3-toolchain`のSB3ソース管理ワークフロー を参照してください。

`app/`から編集用SB3を生成します。

```
pnpm sb3:build
```

生成時に、Title背景の`Version <version> (YYYY/MM/DD)`へ`package.json`の versionとAsia/Tokyoのビルド日を自動で埋め込みます。同時に、公式Webサイトボタンへ Webサイトのブランド画像の正本である`site/favicon.png`を埋め込みます。`app/`には プレースホルダーを保持し、一時ソースで2つのSVGの内容、MD5、`assetId`、`md5ext`、 archive entryを同時に更新するため、ビルドで正本は変更されません。

過去のリリースを同じ日付で再現するときは、日付を`YYYY-MM-DD`で明示します。不正な 日付はエラーにし、暗黙に補正しません。

```
KAMISHIBAI_BUILD_DATE=2026-07-31 pnpm sb3:build
```

`tmp/kamishibai.sb3`をTurboWarpで編集し、別の明示的なパスへ保存してから取り込みます。

```
pnpm sb3:import -- /path/to/edited-kamishibai.sb3
git diff -- app
pnpm sb3:check
pnpm test
pnpm run build
```

`app/project.source.json`は次の不変条件を維持します。

- 浦島太郎などの公開サンプル固有ターゲット、画像、音声を含めない
- 台本解析・実行用リストを空の初期状態で保持する
- 組み込み台本用の予約変数を一意に保持する

展開形式とアセット・拡張の整合性は`pnpm sb3:check`で検証します。toolchainの共通 検証項目を本ガイドへ重複して列挙しません。

DSL、Loading表示、入力、分岐、テキスト、画面遷移などの振る舞いを変更するときは、 同じPRでDSL資料、コマンド資料、 内部仕様書、VMまたはブロック構造のテストを更新します。

8. 埋め込み機能拡張を更新する

`app/extensions/`のJavaScriptは同期済み成果物です。バグ修正や機能追加は、 `app/embedded-extensions.json`に記録された上流リポジトリで行い、レビュー済みの 成果物だけを本リポジトリへ取り込みます。

`status`、 `sync`、 `update`の意味、由来情報の形式、transactionalな更新、ID移行の 対象schemaは、 `sb3-toolchain`のSB3ソース管理ワークフローと `埋め込み拡張IDの移行` を正本とします。ここでは紙芝居アプリへ反映する手順だけを示します。

```
pnpm sb3:extensions:status
pnpm sb3:extensions:sync
pnpm sb3:extensions:update -- EXTENSION_ID
```

上流で拡張IDが変更された場合は、toolchainのID移行を伴う更新を使います。

```
pnpm sb3:extensions:update -- OLD_ID --migrate-id NEW_ID
```

上流の成果物パスも変わった場合だけ`--artifact PATH`を追加します。更新後は必ず `git diff -- app`、 `pnpm sb3:check`、 `pnpm test`、 `pnpm run build`を確認し、 生成したSB3をTurboWarpで開いて対象拡張の主要機能を確認します。

9. ビルダーの実装を変更する

公開APIは`src/builder/index.js`、CLIは`src/builder/cli.js`と`bin/`、仕様テストは `test/builder.test.mjs`にあります。公開API、CLI、アセットマニフェスト、決定的生成、 transactional更新の現行仕様は「SB3・台本変換ビルダーを利用する」（第6章）を 参照してください。この章では仕様そのものを繰り返さず、実装変更時の確認事項だけを 扱います。

APIまたはCLIを変更するときは次を同じ変更に含めます。

- API入力、返り値、エラーの後方互換性の判断
- CLIの`--help`と引数検証
- アセットマニフェストと出力manifestの形式
- 決定的生成とtransactional更新のテスト

- 内部仕様書のAPI／CLI例
- 破壊的変更の場合は新しいメジャーバージョン

```
node --test test/builder.test.mjs
node bin/tmpose-kamishibai.mjs --help
pnpm typecheck
pnpm pack:check
```

10. ドキュメントとサイトを変更する

一般文書は`docs/general/`、体験会資料は`docs/workshops/<日付>/`、公開入口は`site/`を 正本とします。

```
pnpm run preview:docs
pnpm run preview:workshop
pnpm run preview:staff
```

子供向け概要書と参加者向け体験会資料だけにrubyganaを適用します。確認する学年を 変更する場合は1から6を指定します。

```
RUBYGANA_GRADE=4 pnpm run build
```

`pnpm run build`は一般文書ごとにWeb Publicationを構築し、Vivliostyle CLIの`toc`設定で `h2`・`h3`までを含む目次を生成します。目次をMarkdownへ重複して記述しません。HTML、 Vivliostyle Viewer、PDF、文書横断目次、画像参照、しおり、favicon、ライセンス、配布SB3を まとめて検証します。Markdownだけを確認して完了にせず、生成されたHTML/PDFも確認します。

Markdownの見出しには章・節番号を書きません。`h1`は番号なしの文書名、`h2`と`h3`は 本文と目次で自動採番します。用語集などの前付けを採番しない場合は、見出しへ `{.unnumbered}`を付けます。本文から見出しを参照するときは、番号ではなく意味の変わらない IDを付けてリンクし、組版結果にも現在の章・節番号を表示する場合は、リンクへ `{data-ref="chapter"}`または`{data-ref="section"}`を付けます。

11. 変更を検証する

11.1 変更対象ごとのテスト

変更対象	主なテスト
builder API／CLI	`test/builder.test.mjs`
展開SB3の構造	`test/sb3-project.test.mjs`、`test/skip-mode.test.mjs`
内部仕様書の構造一覧	`test/internal-specification.test.mjs`
TurboWarp実行結果	`test/turbowarp-vm.test.mjs`
入力、分岐、wait	`test/async-input.test.mjs`、`test/register-branch.test.mjs`、`test/wait-action.test.mjs`
文書、画像、ライセンス	`test/docs-config.test.mjs`、`test/docs-images.test.mjs`、`test/documentation-license.test.mjs`
公開物と汎用性	`test/sb3-publication.test.mjs`、`test/build-freshness.test.mjs`

11.2 標準チェック

```
pnpm lint
pnpm format
pnpm typecheck
pnpm test
pnpm run build
```

GitHub Actionsはcleanなlinux環境で`pnpm install --frozen-lockfile`、`pnpm test`、`pnpm build`を実行します。ローカルで成功しても、未追跡ファイルや既存生成物へ依存して いないことをCIで確認します。

`pnpm run build`は少なくとも次を生成・検証します。

- `dist/downloads/kamishibai.sb3`
- 一般文書ごとのWeb Publication、HTML/PDF、Vivliostyle CLIが`h2`・`h3`までから生成する目次
- 参加者向け・スタッフ向け体験会資料
- 公開サイトのリンク、画像、目次、PDF bookmark、favicon

SB3またはruntimeを変更した場合は、生成SB3をTurboWarpで開いて次を手動確認します。

- 読込エラーがない
- green flagでtitleとmenuが表示される

- 外部台本と組み込み台本の対象フローが開始できる
- pause、Space、Right、Downの進行が意図どおり動く
- Downでシーンを飛ばしてもBGMが継続し、残りのtransitionが最終状態になる
- Loading、画像、音声、テキストが正しく表示・再生される

内部構造を変更したPRでは、`app/project.source.json`と内部仕様書のtarget、変数、message、 hat、custom block一覧を同時に更新します。

12. 公開する

12.1 GitHub Pages

```
pnpm run deploy
```

`predeploy`がフルbuildを行い、成功した`dist/`だけを`gh-pages`へ公開します。公開後は top page、文書一覧、各カードのHTML／Vivliostyle Viewer／PDF、SB3 downloadを実際の URLから確認します。

問題がある場合は、直前の検証済みcommitをcheckoutしたcleanな環境から再度build・ deployします。生成済み`dist/`だけを手作業で修正しません。

12.2 npmパッケージ

公開済みversionは変更・再利用できません。releaseごとに新しいversionとGit tagを使います。

1. `package.json`、lockfile、`src/builder/constants.js`、READMEの導入例を同じversionへ更新する。
2. cleanなcommitで標準チェック、フルbuild、公開内容のdry-runを実行する。

```
pnpm release:check
```

3. tarballのファイル一覧、license、size、CLI/APIを確認する。
4. Git worktreeではなく通常のclean cloneから公開する。
5. WebAuthnなどの認証を完了してpublic packageとして公開する。

```
npm publish --access public
```

6. registry反映後にmetadataを確認する。

```
npm view @kubohiroya/tmpose-kamishibai@<VERSION> \
  version license dist-tags.latest dist.integrity --json
```

7. 一時ディレクトリへ公開版を導入し、CLIの`--version`と`@kubohiroya/tmpose-kamishibai/builder`のimportを確認する。
8. 公開に使った確定commitへannotated tagを作り、GitHub Releaseを作成する。

公開後に問題が見つかった場合は対象versionを`npm deprecate`し、修正版を新しいpatch versionとして公開します。公開済みtarballやtagを差し替えません。

13. 開発上の問題を解決する

症状	確認と対応
`sb3:import`が置換を拒否する	`git status`と`git diff -- app`を確認し、toolchainの手順に従う
`.app.rollback-*`や`.＜出力名＞.rollback-*`が残る	削除前に元出力と比較し、toolchainの失敗時の扱いに従う
埋め込み拡張が追跡refと異なる	`pnpm sb3:extensions:status`で確認し、固定commitへ戻すなら`sync`、更新するなら`update`を使う
PDF生成browserが見つからない	Chrome/Chromiumを導入し、必要なら`VIVLIOSTYLE_CHROME_PATH`を設定する
ローカルだけtestが通る	生成物と未追跡ファイルを確認し、clean cloneと`pnpm install --frozen-lockfile`で再現する
builderが既存出力を更新しない	エラーの`stage`、asset名、URIを確認する。rollback領域が残っていないか確認する
公開直後にnpm registryが404になる	同じversionを再publishせず、npm公開pageとregistryの反映を待って確認する

復旧でGit履歴を破壊しません。公開済み変更は`git revert`または新しい修正PRで戻し、tagを 移動しません。

14. ライセンスと秘密情報を扱う

対象	ライセンス
`docs/general/**`	CC BY-SA 4.0
`docs/workshops/**`	Copyright © 2026 Hiroya Kubo. All rights reserved.
上記以外で個別表示のない、本プロジェクトが著作権を持つソフトウェアと素材	MPL-2.0

詳細は`LICENSES.md`、`docs/general/LICENSE.md`、`docs/workshops/LICENSE.md`を参照してください。

第三者の画像、音声、font、model、機能拡張には個別のlicenseまたは利用条件が適用されます。 builderで組み込む素材はasset manifestの`license`へ由来を記録します。許諾が確認できない 素材を本体またはsampleへ追加しません。

token、npm認証情報、秘密鍵、個人情報をrepository、SB3、台本、manifest、生成HTMLへ 記録しません。認証情報は環境変数、OSのkeychain、GitHub Secretsなど、公開物へ含まれない 仕組みで渡します。

15. 関連プロジェクトを確認する

TMPose紙芝居の開発から分離し、他のTurboWarp作品や開発環境でも利用できるものを 各リポジトリで公開しています。各プロジェクトの仕様、開発手順、リリースはリンク先を 正本とします。

15.1 sb3-toolchain

`sb3-toolchain`は、SB3をGit差分可能な 展開ソースとして管理し、検証して決定的に再構築するためのCLI／JavaScript APIです。 このリポジトリでは固定依存として利用し、`app/`のimport、検証、build、埋め込み 機能拡張の同期とID移行を担います。

- SB3ソース管理ワークフロー
- SB3展開ソース形式 v1
- 埋め込み拡張IDの移行

15.2 Viteプラグイン

- `vite-plugin-turbowarp-extension`: TypeScriptプロジェクトを単一ファイルのTurboWarp機能拡張としてbuildするViteプラグイン

15.3 TurboWarp 機能拡張開発用テンプレート

- `turbowarp-extension-template`: Viteを使ったTurboWarp機能拡張の開発、テスト、build、リリース用テンプレート

15.4 TurboWarp 機能拡張

- `turbowarp-tmpose`: Teachable Machine Poseモデルを利用したカメラ姿勢認識
- `turbowarp-text-lines`: テキストの行数取得、行単位の読み出し・分割
- `turbowarp-asset-manager`: IndexedDBとSB3内の画像・音声を扱うアセット管理
- `turbowarp-async-input`: キーボード、ポインター、姿勢入力を対象ごとに扱う非同期入力
- `turbowarp-runtime-expression`: runtime変数を使う条件式の安全な評価とbroadcast監視

15.5 その他のライブラリ

- `rubygana`: 日本語テキストの読み仮名を生成するNode.jsライブラリ

16. 関連ドキュメントを確認する

- `03-user-guide.md`：アプリの利用方法と成果物の使い分け
- `04-dsl-manual.md`：台本の構造と書き方
- `05-command-reference.md`：コマンドとアクションの仕様
- `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`：DSL とアプリの変更履歴
- `README.md`：プロジェクト全体の入口と主要コマンド