Busy Tag

.busytagdesign File Format Specification

Schema version 1

Document revision: April 2026

SIA BUSY TAG

Abstract

`.busytagdesign` is the portable container the Busy Tag application uses to share user-created designs between devices. A bundle holds one or more design entries: each entry pairs a rendered `.png` or `.gif` image with the metadata needed to reproduce the design on a recipient's Busy Tag (text overlay parameters, LED color and brightness, display name, creation timestamp). This document specifies the on-disk format, the JSON manifest schema, the validation rules the importer enforces, the `busytag://design` deep-link used for share-by-link, and the HTTP endpoints that host shared bundles.

Contents

1 Container **1**

2 Internal layout **1**

3 `design.json` — single-design bundle **2**

4 `design.json` — collection bundle **2**

5 Field reference **3**
5.1 Top-level (`DesignBundleManifest`) . . . 3
5.2 Per-design entry . . . 3
5.3 `textImageParameters` (`TextImageSettings`) . . . 4

6 Import validation rules **4**

7 Versioning **5**
7.1 Current version . . . 5
7.2 Compatibility . . . 5

8 Deep-link URL **5**

9 Backend share endpoints **5**
9.1 POST /designs/share . . . 5
9.2 GET /designs/shared/{token} . . . 6
9.3 DELETE /designs/shared/{token} . . . 6
9.4 Lifecycle . . . 6

10 Producer notes **7**

11 Consumer notes 7

1 Container

A `.busytagdesign` bundle is a standard ZIP archive.

- **File extension:** `.busytagdesign`
- **MIME type:** `application/vnd.busytag.design`
- **Magic bytes:** `PK\x03\x04` (standard ZIP local-file header)
- **Maximum size:** 50 MB (`DesignBundleService.MaxBundleSize`). Inputs larger than this fail import with the `TooLarge` error.

The container is intentionally a plain ZIP so that any mainstream unarchiver can inspect or repack a bundle for debugging. Production producers should never include directory entries or path components inside the archive — all entry names must be bare filenames (see §6).

2 Internal layout

```
mybundle.busytagdesign
+- design.json        (manifest, always present)
+- <imageFile>        (one or more PNG/GIF entries)
```

- Exactly one `design.json` entry; an importer that does not find it returns the `Malformed` error.
- One image entry per design. For a single-design bundle there is exactly one image; for a collection bundle (§4) there is one image per entry in the `designs` array.
- Image filenames must be unique within the archive. The producer suffixes `_1`, `_2`, etc. if two designs would collide on the same source filename.
- The manifest is written with `CompressionLevel.Optimal` (small JSON benefits from deflate). Images are written with `CompressionLevel.NoCompression` because PNG/GIF are already compressed and re-deflating them only burns CPU.

3 `design.json` — single-design bundle

A single-design bundle has `kind == "design"` and uses the top-level metadata fields. The `designs` array is omitted (or `null`).

```
{
  "schemaVersion": 1,
  "kind": "design",
  "appVersion": "1.5.9",
  "name": "Sunset",
  "imageFile": "sunset_8a3kf2lq.png",
  "ledColor": "FF8800",
  "ledBits": 127,
  "createdTimestamp": "27/04/2026 14:32:10",
  "textImageParameters": {
    "imageSource": "",
```

```
    "text": "Hello",
    "textColor": "#FFCCCCFF",
    "fontSize": 28,
    "textAlign": 1,
    "fontFamily": "Arial",
    "backgroundColor": "#FF8B008B",
    "isBoldText": false,
    "isItalicText": false,
    "isUnderlineText": false,
    "paddingLeftRight": 5,
    "paddingTop": 5,
    "textVerticalAlign": 0,
    "imageScaleMode": 0
  },
  "designs": null
}
```

4 `design.json` — collection bundle

A collection bundle has `kind == "collection"` and lists each design as an entry inside the `designs` array. Top-level per-design fields (`name`, `imageFile`, `ledColor`, etc.) are present at their default values but are ignored on import.

```
{
  "schemaVersion": 1,
  "kind": "collection",
  "appVersion": "1.5.9",
  "name": "",
  "imageFile": "",
  "ledColor": "FF0000",
  "ledBits": 127,
  "createdTimestamp": "",
  "textImageParameters": { /* defaults */ },
  "designs": [
    {
      "name": "Sunset",
      "imageFile": "sunset_8a3kf2lq.png",
      "ledColor": "FF8800",
      "ledBits": 127,
      "createdTimestamp": "27/04/2026 14:32:10",
      "textImageParameters": { /* ... */ }
    },
    {
      "name": "Coffee time",
      "imageFile": "coffee_time_4xq9lp2c.gif",
      "ledColor": "664422",
      "ledBits": 100,
      "createdTimestamp": "20/04/2026 09:15:00",
      "textImageParameters": { /* ... */ }
    }
  ]
}
```

Robustness note

The importer also accepts a manifest where `kind` is missing or unknown but `designs` is non-empty and the top-level `imageFile` is empty — in that case it is treated as a collection. This makes future producers that forget to set `kind` forward-compatible, at the cost of one extra heuristic.

5 Field reference

5.1 Top-level (`DesignBundleManifest`)

Field	Type	Notes
`schemaVersion`	integer	Currently `1`. Higher values fail import as `UnsupportedSchema`.
`kind`	string	`"design"` (default) or `"collection"`.
`appVersion`	string	Producer's app version. Informational only.
`designs`	array | null	Populated only when `kind == "collection"`.

5.2 Per-design entry

These fields appear at the top level for single bundles and inside each `designs[]` entry for collections.

Field	Type	Notes
`name`	string	Display name. If empty on import, the design is named "Imported design".
`imageFile`	string	*Bare* filename of the image entry inside the archive. Required.
`ledColor`	string	Hex RGB or RGBA (no `#` prefix), e.g. `"FF8800"`. Default `"FF0000"`.
`ledBits`	integer 0–127	Bitmask selecting which of the 7 LEDs the design lights up. Bit 0 is the first LED, bit 6 is the seventh. Examples: `0` = all off, `3` = first two, `7` = first three, `127` = all seven. The device receives this verbatim as the first argument of `AT+SC=ledBits,color`. Default `127`.
`createdTimestamp`	string	`dd/MM/yyyy HH:mm:ss`. Preserved on import when present; otherwise the importer sets it to "now".
`textImageParameters`	object	Full `TextImageSettings` payload (next section).

5.3 `textImageParameters` (`TextImageSettings`)

Field	Type	Default
`imageSource`	string	`""`
`text`	string	`""`
`textColor`	SKColor (`"#AARRGGBB"`)	CornflowerBlue
`fontSize`	float	`28`
`textAlign`	integer (`SKTextAlign`)	`1` (Center)
`fontFamily`	string	`"Arial"`
`backgroundColor`	SKColor	DarkMagenta
`isBoldText`	bool	`false`
`isItalicText`	bool	`false`
`isUnderlineText`	bool	`false`
`paddingLeftRight`	float	`5`
`paddingTop`	float	`5`
`textVerticalAlign`	integer (`TextVerticalAlign`)	`0` (Top)
`imageScaleMode`	integer (`ImageScaleMode`)	`0` (Fill)

SKColor values use the `SkColorConverter` format `"#AARRGGBB"` (hex with leading hash, alpha–red–green–blue order) and round-trip via `SKColor.ToString()` / `SKColor.Parse()`.

6 Import validation rules

The importer (`DesignBundleService.ImportAsync`) enforces these rules in order. Any failure aborts the import with the named error.

1. Bundle size $\leq$ 50 MB, otherwise `TooLarge`.
2. A `design.json` entry exists and parses as JSON, otherwise `Malformed`.
3. `schemaVersion` ≤ 1, otherwise `UnsupportedSchema`.
4. For each entry to materialize (one for single, N for collection):
   (a) `imageFile` is non-empty.
   (b) `imageFile` contains no `/`, `\`, or `..` (zip-slip guard); otherwise `Malformed`.
   (c) Image extension is `.png` or `.gif` (case-insensitive); otherwise `UnsupportedImage`.
   (d) A zip entry with that exact name exists; otherwise `Malformed`.

On success, each image is written to the application's local data directory with a freshly sanitized + 8-character random suffix to avoid collisions with existing designs. A new `BusyTagImageModel` is appended to the user's design list and `DesignsImageList.json` is saved.

7 Versioning

7.1 Current version

`schemaVersion = 1`. The `kind` discriminator and the `designs[]` array (collection bundles) are part of the initial released schema.

7.2 Compatibility

- Reading: an importer rejects bundles whose `schemaVersion` is *higher* than the version it understands. Lower or equal versions are accepted.
- Producers should always emit the highest version they support.
- Adding new optional fields is a non-breaking change. Removing or renaming fields, or repurposing existing ones, requires a version bump.

8 Deep-link URL

The Share-via-link flow uploads a bundle and yields an opaque token. The recipient opens a URL of the form:

```
busytag://design?v=1&t={token}
```

Param	Type	Notes
`v`	integer	URL schema version. Currently `1`.
`t`	string	Share token, characters `[A-Za-z0-9_-]` only. Issued by the backend (§9); typically 12 base64url characters from 9 random bytes.

The handler (`DeepLinkHandler.ProcessDesignShareLink`) validates the URL, calls `IDesignShareTrans` to fetch the bytes, then forwards the stream to `DesignBundleImportHandler.HandleStreamAsync` so the same import path used for file-association opens applies.

9 Backend share endpoints

The Busy Tag mobile/desktop app uses three anonymous endpoints under `https://greynut.com/api/busytag/designs` for share-by-link. None require an `X-Device-Key` header; they are open to any caller.

9.1 POST /designs/share

Upload a `.busytagdesign` bundle. Multipart form-data, single field `bundle`.

```
POST /api/busytag/designs/share HTTP/1.1
Content-Type: multipart/form-data; boundary=...

--...
Content-Disposition: form-data; name="bundle"; filename="bundle.
   busytagdesign"
Content-Type: application/vnd.busytag.design

<binary zip body>
--...--
```

Response 200:

```
{
  "token":       "ab3kf9xq",
  "revokeToken": "rk_8h2ng7vd...",
  "expiresAt":   "2026-05-27T00:00:00+00:00"
}
```

Errors: `400` (missing field, magic-byte check fails, invalid zip), `413` (>50 MB), `500` (storage or DB error). The 500 response includes the underlying error message in the `error` field.

9.2 GET /designs/shared/{token}

Stream the previously-uploaded bundle.

```
GET /api/busytag/designs/shared/ab3kf9xq HTTP/1.1

200 OK
Content-Type: application/vnd.busytag.design
Content-Length: 412330
Content-Disposition: attachment; filename="ab3kf9xq.busytagdesign"

<binary zip body>
```

Errors: `404` (unknown token, expired, or the file backing the row is missing).

9.3 DELETE /designs/shared/{token}

Revoke a previously-issued share. The caller proves ownership via the `X-Revoke-Token` header returned by the upload response. Token comparison uses `hash_equals` on the server.

```
DELETE /api/busytag/designs/shared/ab3kf9xq HTTP/1.1
X-Revoke-Token: rk_8h2ng7vd...

204 No Content
```

Errors: `400` (header missing), `403` (wrong revoke token), `404` (unknown token).

9.4 Lifecycle

- TTL: 30 days from upload time (`SHARED_DESIGN_TTL_DAYS`). The server prunes expired rows + files via a probabilistic cleanup that runs on approximately 2 % of all API requests.
- Storage: bundles live as `/var/www/html/api/busytag/shared_designs/{token}.busytagdesign`
- Schema: `busytag_shared_designs` table holds `token`, `revoke_token`, `file_path`, `file_size`, `expires_at`, `created_at`.

10 Producer notes

- Always set `schemaVersion = 1` and an explicit `kind` field, even for single-design bundles.
- Image filenames inside the zip should be safe identifiers — ASCII letters, digits, underscore, hyphen, dot — because some clients surface them in user-visible error messages.
- Strip absolute paths from `imageFile`. The importer rejects anything containing `/`, `\`, or `..`.
- Keep collection bundles under 50 MB. Practically this caps a collection at ~50–100 typical designs.
- The producer in this app's source (`DesignBundleService`) is the reference implementation.

11 Consumer notes

- Do not assume `.busytagdesign` extension. Some share targets rewrite the extension to `.zip` when the type association is not registered system-wide. The Busy Tag importer validates by manifest content; mirror that behaviour.
- Treat the manifest's `appVersion` as opaque informational metadata.

- Never trust the on-disk filename or path inside the archive — always rewrite to a safe, locally-unique filename when extracting, as the reference importer does.