Partial Import / Export (“Bundles”) – Product Brief (Draft)

This document accompanies:

- `docs/2026 - april release x/partial-import-export.requirements.md`
- `docs/2026 - april release x/partial-import-export.technical-spec.md`

It explains the feature in non-technical terms: what it solves, how it behaves, and why it’s a safe step forward while the app still uses a single database.

---

1. What We’re Solving

Right now the app behaves like “one big library”:

- Everything you’ve made lives in one database.
- Export/import is effectively “all or nothing”.

That’s a problem because the most common sharing use-case is not “here’s my whole library”. It’s usually:

- “Here’s my **collection** of cards for this quest/campaign.”
- “Here are **a few cards** I want to share with a friend.”
- “I want to bring someone else’s set into my library without overwriting my own work.”

Partial Import/Export fixes this by introducing **Bundles**: a way to package and share only part of your database safely.

---

2. The Core Idea: Share a Collection as a Bundle

A **Bundle** is a single file that contains:

- One or more collections
- The cards inside those collections
- The images those cards depend on

Think of it like a “deck pack” you can send to someone.

---

3. The Most Important Principle: Merge by Default (Safe)

Importing a bundle should not feel risky.

By default, importing a bundle **merges** into your current library:

- It adds what you don’t already have.
- It avoids duplicating identical cards and images.
- If there’s a real conflict, it asks what you want to do.

There is also an explicit “danger zone” option to **replace** your database (full overwrite), but that’s not the primary value of this feature.

---

4. What “Smart Merge” Means (In Human Terms)

When importing, the app should behave like a careful assistant:

4.1 Images: don’t store duplicates

If the imported bundle contains an image you already have, the app should reuse the existing image rather than storing another copy.

Result: smaller database, fewer duplicates, and imported cards still look correct.

4.2 Cards: avoid duplicates, but don’t hide conflicts

Imported cards can match what you already have in a few ways:

- **It’s already here (identical)**: the app shouldn’t create a duplicate.
- **It looks identical but has a different internal ID**: the app should treat it as a duplicate and reuse the existing card by default.
- **It has the same internal ID but different content**: that’s a true conflict. The app must ask:
- Keep mine
- Replace with imported
- Keep both

This keeps importing safe, predictable, and transparent.

---

5. A Safety Feature That Matters: “Imported: …” Collection

After importing, the app creates a new collection like:

- `Imported: <bundle name>` (or a date)

This is a practical safety rail:

- You can review what came in.
- You can keep it separate from your existing sets.
- You can later move cards into other collections if desired.

---

6. What This Enables (User Stories)

Bundles unlock a bunch of user-friendly workflows:

- Share a collection with a friend without exporting your whole library.
- Import someone else's collection without overwriting yours.
- Trade community-made sets as single files.
- Maintain one database for now, but behave as if you have "mini projects" by moving sets around via bundles.

This directly addresses the current "all or nothing" limitation of a single database.

---

7. How It Fits the Roadmap

Bundles are a stepping stone toward richer organization later (e.g., multiple projects/databases), without requiring that bigger feature now.

Even after "projects" exist, bundles still matter:

- Sharing is still valuable.
- Moving data between projects is still needed.

So this isn't throwaway work—it's a foundational capability.

---

8. UX Expectations (Non-Technical)

To feel trustworthy, import should be a guided flow:

1. Choose a bundle file.
2. See a clear preview: how many cards/images will be added, linked, or need attention.

3. Resolve conflicts only if needed.
4. Apply and show a short report of what happened.

The user should always feel in control.

---

9. Summary

Partial Import/Export (Bundles) turns sharing from “all or nothing” into something practical:

- Export a collection as a portable bundle.
- Import it safely by merging and deduplicating.
- Resolve conflicts explicitly when necessary.

It’s a high-value quality-of-life feature that makes the app more collaborative and future-ready while still operating as a single-database tool.