# Proposed IFT Spec Outline

For Static Patches

Author: Garret Rieger
Date: Jan. 16th, 2024

# Proposed Specification Outline

This will replace the existing Patch Subset and Range Request specification text:

1. Introduction
2. Opt-In Mechanism
    a. Fallback
    b. Offline Usage
3. Extensions to the Font Format
    a. Subset Definition
    b. Patch Maps
    c. ‘IFT ‘ and ‘IFTX’ table.
        i. Format 1: Codepoint and feature only mappings.
        ii. Format 2: General mapping.
4. Font Patches
    a. Shared Brotli
    b. Per Table Shared Brotli
    c. Binned Patch
5. Extending a Font Subset
6. Privacy and Security Considerations

# Opt-In Mechanism

- Only one method now, so this can be simplified.
- Will continue to use tech(incremental), can drop tech(incremental-*) keywords.

# Patch Mapping

The mapping will contain a series of mapping entries:

**Subset definition ⇒ Patch URL, Patch Encoding**

Where subset definition is composed of three sets:

1. Code points
2. Feature tags
3. Design space

# Patch Mapping

- Mapping entries will be stored in the font in two tables: ‘IFT ‘ and ‘IFTX’
- Both tables have the same format and the entries from both are unioned together to form the final mapping.
- Having two separate tables allows different patch types to touch their own mapping entries without interfering with the other types (eg. using per table shared brotli patches).

# Patch Mapping: Format 1

- This will essentially be the mapping table defined in the IFTB proposal. Supports:
  - Mapping code point and feature tag subsets.
  - Doesn’t support overlapping subset definitions in the map keys.
  - More compact than format 2.

# Patch Mapping: Format 2

- The specific encoding is still TBD, but this will be a more general purpose mapping that will have more flexibility then format 1, notably:
  - Supports design space
  - Supports overlapping subset definitions in entries.
  - Will be very similar to the prototype schema, but not using protobufs.

# Font Patches

- Define independent and dependent patches.
- Patch types:

| Format | Granularity | Dependent | Notes |
|---|---|---|---|
| Shared Brotli | Font File | Yes | |
| Per Table Shared Brotli | Font Table | Yes | One patch can update multiple tables. Option to replace tables as well. |
| Binned | Glyph | No | IFTC/IFTZ from IFTB proposal |

# Extending a Font Subset

- This section will define the algorithm that a client follows to interpret the mapping tables and apply patches in order to extend a font subset to a new desired subset definition.

# Extending a Font Subset: Algorithm

- Input
  - Font Subset File
  - Target Subset Definition
- Output
  - Extended Font Subset File

# Extending a Font Subset: Algorithm

Step 0: Validate mapping tables

- Check table validity according to rules provided in the previous sections.

# Extending a Font Subset: Algorithm

## Step 1: Patch Selection

1. Find all map entries which intersect the target subset definition. An entry intersects when:

| | **Target Set Unspecified** | **Target Set Specified** |
|---|---|---|
| **Map Entry Set Unspecified** | Match | Match |
| **Map Entry Set Specified** | No Match | Check for Intersection |

   - All sets (codepoint, feature, design space) in the mapping entry must be considered to match by the above rules for the map entry to be considered to match.

# Extending a Font Subset: Algorithm

Step 1: Patch Selection (Continued)

2. If the set of match entries has one or more dependent patches, then pick exactly one of the dependent patches and return it.
    a. The selection algorithm at this step will likely be left unspecified (implementation detail).
    b. The client could opt to pre-fetch independent patches as well at this step, anticipating needing them later.
3. If there are one or more independent patches, return all independent patches.
4. Otherwise, if there are no matched patches, augmentation is finished.

# Extending a Font Subset: Algorithm

Step 2: Fetch and Apply Patches

- Fetch all patches returned in step one, and apply them to the font. Then go back to step 1 using the font subset produced from patch application as the input.

# Privacy and Security

- This should be mostly copied from existing specification text, with some small changes.
- In the patch subset spec we enforced codepoint grouping to help protect privacy. In the new static patching setup this isn’t required since we’re no longer dynamically forming exact sets of content codepoints.