



Automating Missing-Data Classification by Applying FPS IsVisited Metadata Upstream in the Blaise Extraction Pipeline

September 2026

April Beswick

Youhong Liu

Mohammad Mushtaq

Institute for Social Research, University of Michigan

1. Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 10,000 U.S. families. Blaise has been the data collection software since 2001. As of 2021, we shifted to mixed-mode data collection with a conservative 40%-60% split between Web and Computer Assisted Telephone Interviewing (CATI). In the two biennial waves since 2021, most of the sample has shifted to Web mode only. Although this transition has expanded respondent flexibility and improved the front-end data collection efficiency, it has also come with challenges downstream during data cleaning and dissemination. Our most significant complexity has been correctly distinguishing between fields that were legitimately off-route due to programmed instrument skips and fields that were on-route but left unanswered by respondents. In the relational data output, these two conditions are indistinguishable: both appear as generic missing values.

Accurate classification of missing data is essential for downstream processing, longitudinal consistency, and analytic validity. The only source capable of differentiating skip-driven missingness from respondent-driven missingness is the Field Properties File (FPS) file, specifically the *IsVisited* metadata. After moving to a mixed-mode environment, PSID applied this information manually after relational extraction, merging FPS metadata at the cell level across all data-out tables. This approach was labor-intensive, potentially error-prone, and produced inconsistencies across modes.

This paper introduces a redesigned workflow that shifts FPS-based logic upstream, applying *IsVisited* information before relational extraction. By programmatically assigning a Blaise Special-Answer: Refused codes to entire Web cases—or to specific Web sections—prior to extraction, the new process harmonizes CATI and Web outcomes at the source and eliminates manual intervention. The remainder of this paper describes the redesigned workflow, its implementation within the Blaise environment, and its implications for mixed-mode instrument design and data quality.

2. Background and Literature

2.1 Mixed-mode data collection and missing-data challenges

Mixed-mode designs introduce several well-documented challenges that directly affect how missing data should be interpreted. Mode effects arise because interviewer-administered CATI interviews and self-administered Web surveys produce different response behaviors: interviewer presence in CATI reduces item skipping and encourages explicit refusals, while Web respondents more often skip items or abandon sections without providing a clear refusal (Hope et al. 2022). At the same time, metadata gaps in the relational output make it impossible to distinguish between legitimate instrument skips and unanswered on-route questions, because relational tables do not retain information about whether a field was displayed or visited. Only FPS metadata—specifically *IsVisited*—contains the routing information needed to make this distinction. Together, these issues create ambiguity in the relational data and underscore the need for upstream missing-data classification using FPS metadata.

2.2 Blaise Metadata and the Role of FPS Files

In Blaise 5, *IsVisited* is a Boolean field property used to determine whether the respondent navigated to or encountered a field during the interview. The *IsVisited* value is stored in the *BDBX* data file. Through a simple manipulation and extraction script, the *IsVisited* information is extracted from the *BDBX* and output to an *FPS file*, which can then be used as a source of item-level metadata.

Normally, the *IsVisited* system property is only available during the interview session. However, the value can be retained by declaring (or re-declaring) the *IsVisited* property in the *FieldProperties* section

using *TBoolean* as the type. *TBoolean* is a predefined enumeration type in Blaise with two values: *false* (0) and *true* (1). When the respondent navigates to the next page, all fields on the current page that are on the route are considered visited.

The *IsVisited* flag records whether a field was displayed to the respondent during the interview. If a question is skipped due to programmed logic and is not displayed, *IsVisited* is set to *False*. If the question is displayed to the respondent, *IsVisited* is *True*, regardless of whether the respondent provides an answer.

This distinction is crucial in mixed-mode environments. Web respondents can encounter a question and leave it blank without providing an explicit refusal. It is important to note that not providing on-screen *DK* (*Don't Know*) or *RF* (*Refusal*) options for Web respondents was a purposeful design decision made by the methodologists and Principal Investigator team when the mixed-mode instrument was designed in 2021. CATI interviewers, on the other hand, cannot leave items unanswered and must record refusals directly using the special-answer hot keys.

2.3 Prior approaches to missing-data CATI/Web harmonization

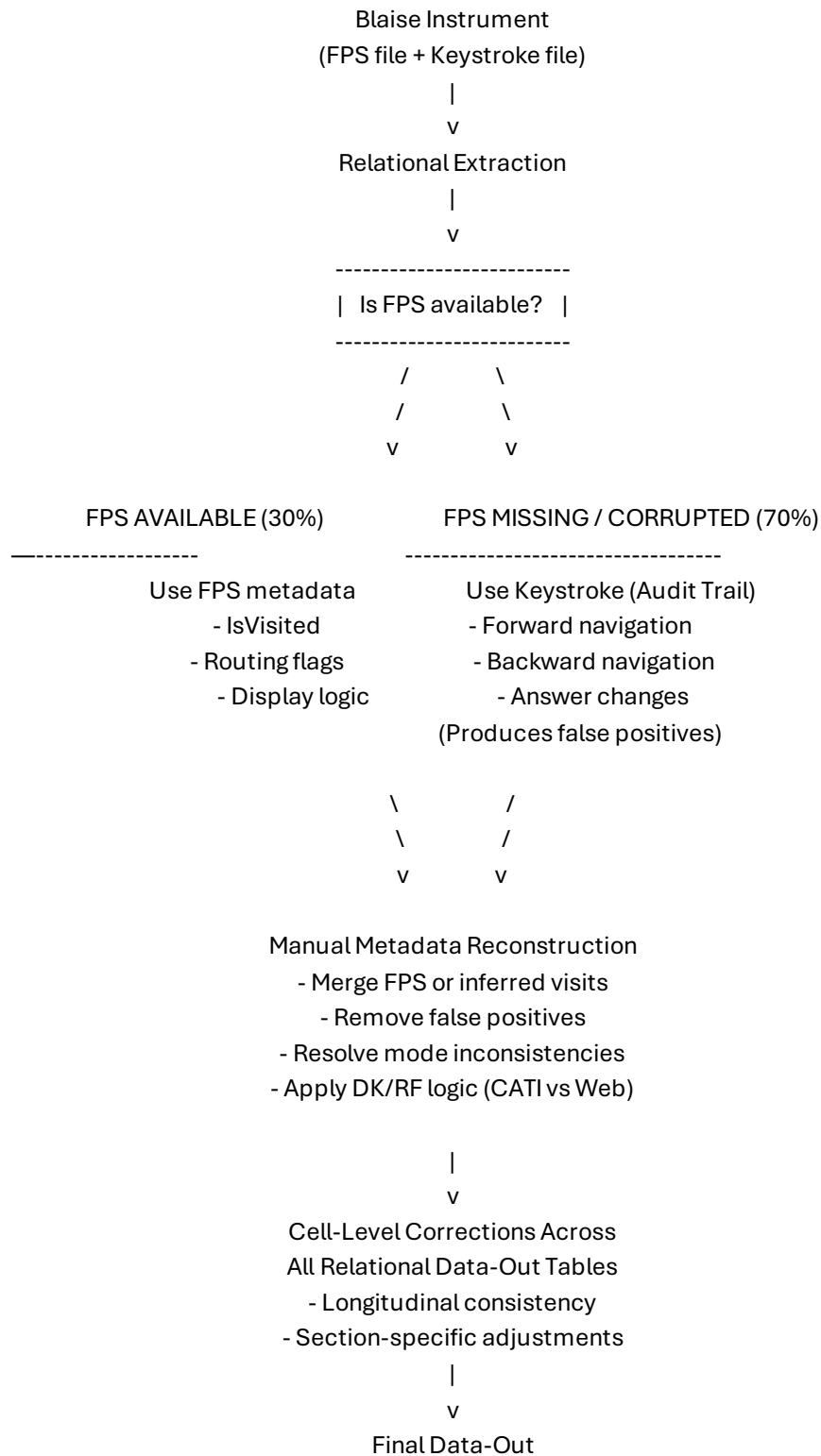
In 2021, our planned approach was to use the FPS file information and apply it post-extraction. Our standard extraction method is not a single wide-file but rather, in ascii-relational tables for each Blaise block. We tested this method in the mixed-mode pilot and it appeared to be the solution we needed. Once we moved to production however, there were several stumbling blocks to the smooth application of this method.

First, we discovered well into production that the FPS file was missing/corrupted for many cases; specifically, **6425** out of **9244** interviews or **70%**. Because the FPS file is the only authoritative source for *IsVisited*, routing flags, and display logic, its absence forced us to rely on the keystroke (audit-trail) file to reconstruct question routing/*IsVisited*. The keystroke file captures every action taken during the interview—including forward navigation, backward navigation, revisiting screens, and overwriting prior answers. While this file contains enough raw information to infer whether a question was ever displayed, it also produces numerous false positives. For example, respondents who navigated backward or changed answers triggered additional audit-trail entries that made it appear as though certain fields had been visited multiple times or visited when they were not part of the final routing path. Staff had to manually correct these false positives, adding another layer of cell-level operations and increasing the labor intensity of the old process.

Secondly, with the breadth and complexity of actual production data once it was extracted, we found that attempting to apply the FPS metadata information to extracted relational tables proved to be quite complex and challenging as the fully extracted loop names do not 100% map directly to the FPS file output. Our goal in the end was to categorize the on-route not answered in the Web mode to the same NA/Refused category in the CATI mode. This was critically important for us as a panel survey in order to harmonize fields across waves and provide consistent codebook values as we added the Web mode (Mushtaq, M., & Beale, A. 2020).

As we have moved to a greater percentage of Web interviews in the subsequent waves, we were looking for a more efficient method to map the *IsVisited* properties to the interview data within our process pipeline which would automate the process and remove manual interventions and corrections.

Figure A: Description of the FPS application process required in 2021



3. Redesigned Workflow: Applying *IsVisited* Upstream

3.1 Conceptual Overview

The redesigned workflow shifts missing-data classification to the FPS stage, before any relational extraction occurs. Because the FPS file reflects the respondent's *final* routing path—and intentionally excludes artifacts created by audit-trail backtracking—it provides the most reliable record of whether each question was displayed. Evaluating *IsVisited* and mode-specific flags while the instrument structure is still intact, allows the system to distinguish between legitimate programmed skips and on-route items that were shown but left unanswered.

Anchoring missing-data logic in FPS metadata removes the ambiguity created by Web mode's lack of explicit DK/RF options and eliminates the need for post-hoc cell-level corrections. The FPS file becomes the authoritative source for routing and display status, enabling a standardized and reproducible missing-data framework across modes and waves of collection.

3.2 Assigning Refused Codes Programmatically

During 2021 development, we initially explored introducing a new special-answer category for Web nonresponse. However, this approach would have required additional width-correction procedures to maintain backward compatibility with 50+ waves of legacy PSID data. Because Blaise automatically assigns DK/RF values wider than our established codebook ranges, adding a new special-answer would have reintroduced the same harmonization challenges we resolved during the Blaise 5 transition.

Instead, we adopted a simpler and more robust strategy: use *IsVisited* to identify on-route, unanswered Web items and assign the existing Refused code directly. This aligns Web behavior with CATI interviewer-recorded refusals without modifying field widths or introducing new special-answer categories.

Specifically, this is how this is achieved in our new process:

Figure B: Include Default FieldProperties to produce the FPS file

1. In the data model, we'll include these default FieldProperties. Then a very simple Blaise to ASCII script Manipula script will produce the FPS file

```
DATAMODEL FES27 "FES27"
{ For Prod version:
DATAMODEL FES25 "FES25 (V1.00)"
}

LANGUAGES    = Eng  "English",
              Span "Spanish"

MODES        = SELFAdmin,
              IWERAdmin

FIELDPROPERTIES
  IsVisited   : (No (0), Yes (1))
  Remark      : OPEN
```

2. During the C# process, if a field is on route and has an empty value, the program attempts to assign **RF** to the field. However, if **RF** is not an allowed value for that field, the assignment fails. The log file will indicate this. For example:

2026-01-13 15:06:20 - 0023033 [NA] String: Section_D.D25_D69[1].D66.D66_CITY - ***RF can be assigned to this field and most fields behave this way.***

2026-01-13 15:06:20 - 0023033 ***Error*** [NA] Specified argument was out of the range of valid values.

Parameter name: value: Section_D.D25_D69[1].D66.D66_ST

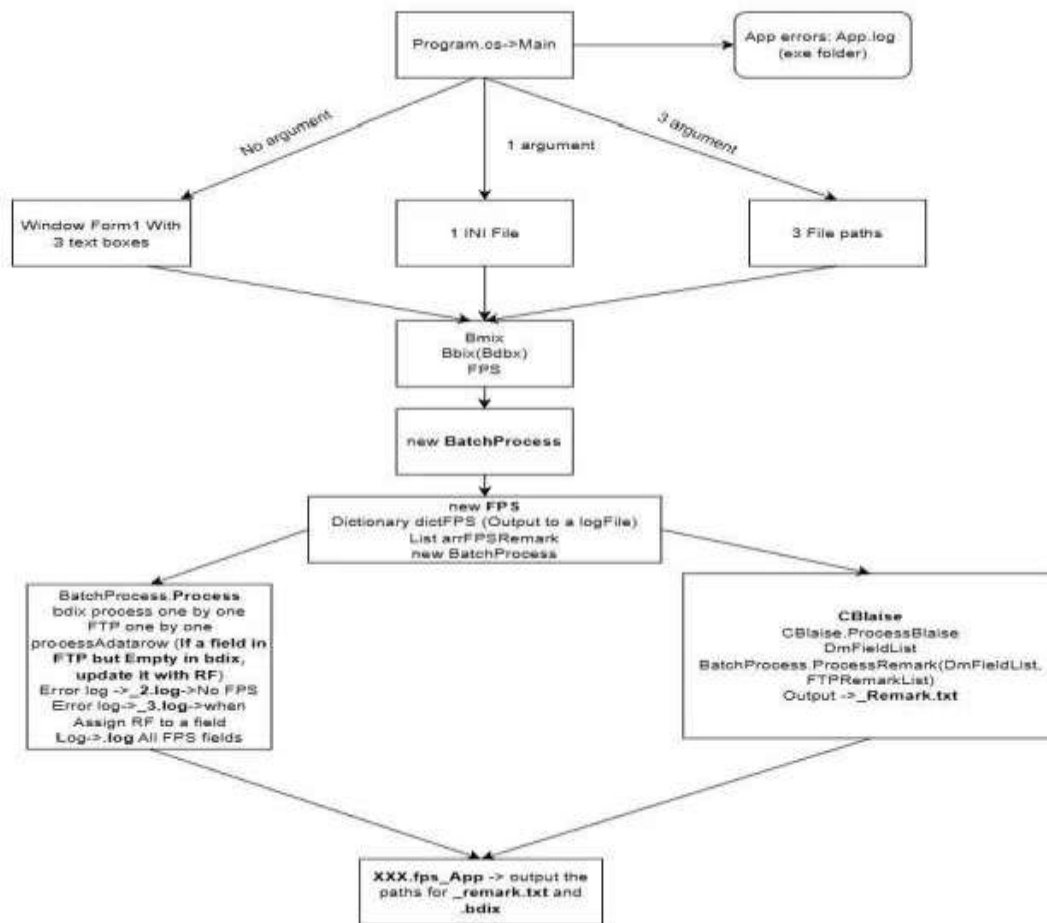
In this example, **Section_D.D25_D69[1].D66.D66_ST** is the field causing the problem because **RF** is not a valid value for that field.

Operationally, the C# process attempts to assign Refused to any on-route field with an empty value. If Refused is not a valid value for a particular field, the assignment fails and is written to the log file for review. In practice, only a small number of fields have *NoDKRF* attributes (e.g. State look-up lists), and these can be handled individually. On the whole, this process works quite seamlessly in our test of our smaller supplemental study Transition into Adulthood Supplement.

3.3 Integration with Blaise Extraction Pipeline

The key architectural change is inserting FPS-based missingness logic *upstream*, before ASCII-to-relational extraction. At this stage, the system evaluates *IsVisited*, routing flags, and mode indicators while the full instrument hierarchy is still available. Missing-data classifications are assigned directly to the Blaise data model, ensuring that each field carries its correct DK/RF status into extraction. By the time relational output is produced, all missing-data logic has already been applied, and no additional metadata merging is required.

Figure C: New FPS Application work-flow



3.4 Automation and Efficiency

By moving missingness classification into the FPS stage, the workflow removes the manual steps that previously occurred after relational extraction. This allows us to not only reduce the time needed from the Applications Development team to apply the cell-level corrections, it allows the Processing team to shorten the gap between pulling data and transforming it for editing. Importantly, it also allows the processing team to see the actual final values of all Web and CATI cases while in production. Instrument routing errors can more easily be identified and reported back to the instrument design team for any needed corrections to the data model.

4. Results and Conclusion

To evaluate the redesigned workflow, we first implemented it in one of PSID's smaller supplemental studies. This allowed us to test the upstream application of *IsVisited* logic, refine the C# assignment procedure, and confirm that Refused codes could be applied reliably to on-route, unanswered Web items. We also tested—and ultimately abandoned—the idea of introducing a new special-answer category, as

doing so would have required additional width-correction steps to maintain compatibility with 50+ waves of legacy data. The upstream Refused-assignment approach proved simpler, more stable, and fully compatible with existing codebook conventions.

The results of the proof-of-concept were clear. Applying missing-data logic at the FPS stage produced relational output that already contained harmonized DK/RF values, eliminating the need for post-extraction metadata merging or cell-level corrections. CATI and Web cases generally follow the same rules, section-level nonresponse was handled consistently, and ambiguous blanks were no longer misclassified. Because the relational extraction simply carries forward the preassigned values, the Processing team can work directly with final values immediately after data pull, reducing both turnaround time for editing and opportunities to find skip errors and correct the instrument during production.

Our next steps are to call the FPS application steps from a command line as part of the overall extraction process. We have also written out deviations to the RF assignments to the log files (e.g. items where RF is not allowed). We also plan to add additional errors to the log if FPS records are missing for completed cases stored in the BDBX. We will plan to scan the logs for these errors during production in order to catch and fix problems early.

References

- Beaule, A., Liu, Y., & Mushtaq, M. (2012). *Blaise-to-SAS Transformation: A Technical Description of PSID's Data Processing Workflow* (Technical Series Paper No. 12-03). Institute for Social Research, University of Michigan.
- Mushtaq, M., & Beaule, A. (2020). *Field Properties Values: A Tool to Identify and Adjust Missing Data from "Relational" Extraction*. PSID (Technical Series Paper 20-03). Institute for Social Research, University of Michigan.
- Hope, S., Campanelli, P., Nicolaas, G., Lynn, P., & Jäckle, A. (2022). *The role of the interviewer in producing mode effects: Results from a mixed modes experiment comparing face-to-face, telephone and web administration*. *Survey Research Methods*, 16(2), 207–226.