

LLM Agents for CRO Data Processing:

Automating Validation, Troubleshooting, and Communication Loops

Kevin Klicki | Informatics Engineer | Montai Therapeutics

CRO Coordination Is a Moving Target

Inconsistent Returns

CRO data return can be missing data, metadata, can arrive in inconsistent formats.

Costly Intervention

High value FTE time is consumed reformatting data and validating metadata before analysis.

Slow Feedback Loops

Back and forth communications with the CRO extend timelines and delay decisions.

CRO Workflow: How Data Moves

Internal



Metadata



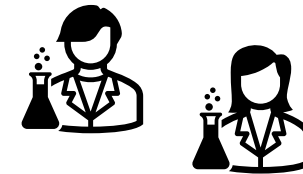
Materials



Insights



CRO



Data/Metadata



CRO Workflow: How Data Moves

Internal

CRO



Metadata

Metadata/Materials mismatch



Materials



Insights



Data/Metadata

CRO Workflow: How Data Moves

Internal



Metadata



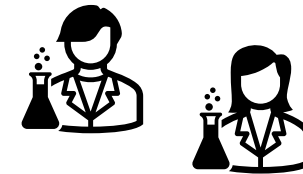
Materials



Insights



CRO



Execution Errors,
Lost Materials



Data/Metadata

CRO Workflow: How Data Moves

Internal



Metadata



Materials



CRO



Malformed Data/Metadata



Insights



Data/Metadata

What's at Stake

FTE Hours Burned

High value scientist and engineer time is spent reformatting and validating CRO returns.

Timelines Extended

Each round of CRO data correction stretches program timelines and pushes back decisions.

Decisions on Bad Data

Worst case: business decisions are made on faulty data and metadata that slipped through.

Mitigation Through Agentic AI

Can agentic AI be deployed to:

- ▶ Validate CRO data returns for proper formatting and complete metadata?
- ▶ Provide actionable feedback to correct validation failures?

Key Outcomes

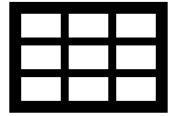
Reduced Turn-Around

From CRO data return to ready-for-analysis, measured in hours not days.

Reduced FTE Burden

Manual format intervention is replaced by self-service guided fixes.

Montai Tx Data Ingress Architecture



Well plate



Tabular



Reports



DUFF

Data Upload
File Formatter



System of Record

Heterogeneous CRO returns are normalized through DUFF before landing in the system of record.

DUFF: From CRO Return to System of Record

What DUFF Does

- ▶ Library of curated parsing functions
- ▶ Inputs: raw CRO data return
- ▶ Outputs: schema compliant data frame
- ▶ Simple web interface built on Django®

The screenshot displays the DUFF web application interface. The top navigation bar includes the title 'DUFF Data Upload File Formatter', a 'New Upload' button, 'My Uploads', 'QC Review', and a user profile 'kklicki@montai.com'. The main content area is divided into two panels. The left panel, titled 'DUFF - Data Upload File Formatter', contains form fields for 'Project' (set to 'Quarantine'), 'Protocol' (set to 'medchem_assessment'), and 'Data Source' (with 'Upload File' and 'SharePoint' buttons). Below these is a file upload section with a 'Drag and drop file here' area, a 'Limit 200MB per file' note, and a 'Browse files' button. At the bottom of this panel are a 'User Guide' link and a large green 'Begin New Data Load' button. The right panel features tabs for 'Data Loading' (active), 'Translation Table', and 'Exclusion Tab'. Under the 'Data Loading' tab, there is a 'Run Metadata' section with a 'Note' field (containing 'New Target Enrichment') and a 'Run Location' dropdown menu (set to 'Montai').

Without Agent Support

User Experience

- ▶ Errors trigger stack dump
- ▶ Not immediately informative
- ▶ Requires engineering intervention

Upload ID: 223Error Type: INTERNAL

Error Message:

'luc_block'

Full Traceback:

Traceback (most recent call last):
File "/Users/kevinklicki/duff_django/apps/uploads/tasks.py", line 1139, in parse_upload_task
dataset.combine_sources(transformed_readouts)
~~~~~  
File "/Users/kevinklicki/duff\_django/apps/datastructure/services/dataset.py", line 93, in combine\_sources  
comp\_data, conver\_table = custom\_parser.combine\_sources(  
~~~~~  
self,
~~~~~  
widget\_keys,  
~~~~~  
)
^
File "/Users/kevinklicki/duff_django/apps/datastructure/services/parsers/pk_parsers.py", line 614, in
combine_sources
'luc_block': input_data.complete_data['luc_block'],
~~~~~  
KeyError: 'luc\_block'

# With Agent Support

## User Experience

- ▶ In-depth explanation
- ▶ Description of expected format
- ▶ Clear resolution steps

### Format Error Report

ADME-MONH-Caco-2 ABBA-20260105-2.xlsx • caco-2\_permeability

#### Issue Summary

The uploaded file "ADME-MONH-Caco-2 ABBA-20260105-2.xlsx" failed to parse correctly due to missing "Virtual MRB" values in the "Compound ID" column.

#### File Analysis

- Present Files: The uploaded file "ADME-MONH-Caco-2 ABBA-20260105-2.xlsx" contains a "Data Summary" sheet as expected.
- Expected Files: The expected file format is an Excel workbook containing a "Data Summary" sheet with specific column headers and data structure.
- Missing/Extra: The "Compound ID" column in the uploaded file contains values that do not match the expected format of "MR-#####-###" or "VMR-#####-###". The error message indicates that the parser could not find a "Virtual MRB" for the compound IDs present in the file.

#### Column Analysis

| Column      | Found                                  | Expected                      | Action Required                              |
|-------------|----------------------------------------|-------------------------------|----------------------------------------------|
| -----       | -----                                  | -----                         | -----                                        |
| Compound ID | "VMR-0002203-001a", "VMR-0002203-001a" | MR-#####-### or VMR-#####-### | Update compound IDs to match expected format |

# Error Classification: Routing to Right Resolution

---

## Engineering

### Infrastructure and Auth Errors

*AWS, database, credentials, system of record submission*

---

### Resolution

Monitoring alerts and user-friendly messages route to the engineering team for investigation.

## AI Agent

### Format and Validation Errors

*Wrong columns, missing data, invalid file structure*

---

### Resolution

Self-service resolution with guided fix steps the user can apply directly in Excel.

# Agent Persona Configuration

## Role: Data Format Quality Assurance Agent

### Specific

Exact cell references, column names, values

### Empathetic

Patient, passive voice for end users

### Actionable

Steps the user can take in Excel

### Thorough

All issues flagged on the first pass, not only issue that caused initial parsing error

## Structured Output

Issue Summary > File Analysis > Column Analysis > How to Fix

# Format Specifications

---

118+

Protocol-Specific  
Parsers

## In Vivo Data

Multi-sheet Excel, multi-tabular conventions

## Tabular Data

Preformatted data fields

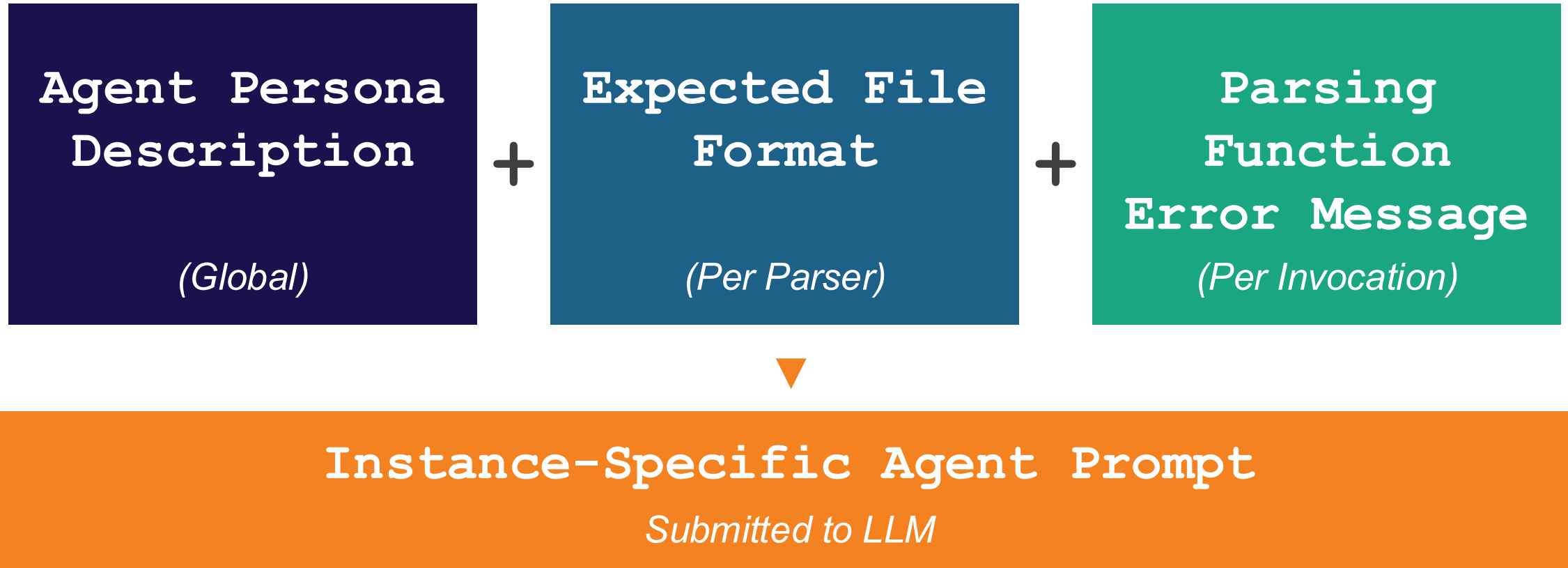
## Plate Based Assays

ZIP archive structure, raw instrument data

*Each parser carries a human/agent-readable spec describing expected columns, structure, and types. Specs are injected into the agent prompt at runtime.*

# Agent Prompt Construction

---



# LLM Implementation

---

AWS Bedrock

Claude 3 Haiku

us-east-1

01

## Initialization

---

Bedrock client defined;  
LLM provider, model, and  
version selected.

02

## Prompt Injection

---

Agent persona passed as  
system argument; file  
format and parsing error  
passed as message  
content.

03

## Agent Response

---

JSON response parsed  
into markdown and  
rendered to the user.

# Agent Output: Issue Summary

## Sample Response

- ▶ Issue summary identifies the protocol mismatch
- ▶ File analysis flags structural problems
- ▶ Concrete next step in plain language

### Format Error Report

20260228\_██████\_Luc\_JK.xlsx • General Tabular Upload

#### 🔍 Issue Summary

The uploaded file appears to be a 384-well plate reader output, not a standard tabular data format expected by the General Tabular Upload parser. Plate reader files have a grid-like structure with row labels (A-P) and column headers (1-24), which is not compatible with the tabular format required.

#### 📁 File Analysis

- Present Files: The uploaded file /tmp/duff\_uploads/20260228\_██████\_Luc\_JK.xlsx is a Microsoft Excel workbook.
- Expected Files: The General Tabular Upload protocol expects standard tabular data with column headers and data rows.

# Agent Output: How to Fix

## Sample Response

- ▶ Column-by-column analysis pinpoints the offending fields
- ▶ How-to-fix instructions reference specific cells and values
- ▶ User can resolve the issue in Excel without engineering

### How to Fix

Since this is a plate reader output file, you will need to use a protocol-specific parser designed for that format, rather than the General Tabular Upload. Here are the steps:

- Identify the specific assay or protocol this plate reader data is intended for (e.g. , CTG, or another assay-specific protocol).
- Select the appropriate protocol-specific parser from the list of available options.
- Upload the file using the correct protocol-specific parser. This will ensure the data is processed correctly according to the expected plate reader format.

### Additional Notes

The General Tabular Upload protocol is designed for standard row-per-record tabular data, not plate reader output. Plate reader files have a unique grid-like structure that requires a specialized parser to interpret correctly. Please make sure to use the appropriate protocol-specific parser for your plate reader data to avoid parsing errors.

# Organizational Impact

---

< 1 day

—  
**Turn-around time**

from CRO data return to  
analysis-ready data in  
system of record

< 1 hour

—  
**FTE time**

spent on data loading  
and manual format  
intervention

# Future State

---

## Expand agentic format handling to agentic parser generation

### 01

---

#### Load & Select

User loads raw data and selects the target data schema.

### 02

---

#### Parse with extant script or generate if error encountered

Agent inspects raw data, infers schematization, and writes a parsing script.

### 03

---

#### Review & Commit

User reviews parsed tabular data before committing to ELN/LIMS.

# Acknowledgements

---

Montai Platform Engineering Team

Montai Biological Sciences Team

AWS Account Reps

# Thank You

---