Paper DH05

Real-time Integration of EDC and Third-Party Data and Query Actions within an R Shiny Data Review and Patient Profile Tool

Pavel Burmenko, Veeva Systems, Seattle, USA
Terek Peterson, Atrous Research, Philadelphia, USA

ABSTRACT

Clinical trial teams increasingly require rapid access to data. This data may be stored in multiple systems and actions related to the data may need to be taken in multiple systems. Clinical data teams may have one tool for collecting data and interacting with site personnel and another tool for visually analyzing the data. Working with multiple tools, there is a need for clear traceability following GxP requirements for audit trails and documented data lineage. Data managers, medical reviewers, statisticians, and programmers can work in tandem by using open source data review tools that are directly integrated with source systems using modern Application Programming Interfaces (APIs) that allow for near-real time access to data and the ability to take actions on that data. This paper will highlight the technical considerations to implement an R Shiny tool with API integration to the source clinical data management system.

INTRODUCTION

Currently, a "Data Lag" exists because Medical Monitors and Statisticians review visualizations and other analytical reports in disconnected SAS output or commercial Business Intelligence (BI) tools, separate from the source data in a clinical data workbench like Veeva Clinical Database (CDB). This solution uses an open-source R Shiny tool integrated directly with the clinical data workbench via modern, incremental APIs to ensure full traceability and direct actionability, such as querying data directly from the visualization. The result provides near real-time data access, direct actionability (querying of data discrepancies), and full traceability compliant with GxP.

SOLUTION ARCHITECTURE

The architecture consists of Veeva CDB as the source system where all clinical data sources are aggregated and structurally validated. Posit Connect is used for security and automation. R Shiny is used for the user interface. The workflow begins with a user logging into Posit Connect via Single Sign On (SSO). The R process then requests Study File Format (SFF) packages from the Veeva API, which are processed into visualizations. The SFF can be consumed as full data extracts or incremental data extracts available every 15 minutes.

In this architecture, Posit Connect serves as the central orchestration layer, providing enterprise-grade security for clinical trial reviewers and automating the complex Extract, Transform, Load (ETL) processes required for near real-time data access.

USER AUTHENTICATION AND SECURITY

Posit Connect ensures that only authorized personnel can access sensitive clinical data by integrating with existing corporate identity providers.

- SSO: Connect integrates with systems like LDAP, Okta, or Active Directory. This allows clinical users to log in with their standard corporate credentials, ensuring a seamless and secure entry point.
- Access Control Lists (ACLs): Permissions are managed at the application level. Administrators can restrict a specific R Shiny app to a defined subset of users, such as "Study A Medical Reviewers".

- Auditability: Every user visit and action is logged. Connect records who accessed the app, when, and for how long, which is critical for maintaining GxP compliance.
- Credential Management: To avoid hard-coding sensitive Veeva API credentials, Posit Connect stores them as Environment Variables. These are encrypted and passed to the R process only at runtime, keeping secrets out of the source code.

JOB AUTOMATION

To solve the "Data Lag" problem, Posit Connect automates the background tasks needed to keep clinical visualizations up to date.

- Scheduled ETL Scripts: Rather than fetching large clinical datasets every time a user opens the app, Posit Connect runs a non-interactive background R script on a schedule (e.g., every 15 minutes).
- Efficient Data Storage (Pins): The background script identifies new incremental packages from Veeva, processes them, and saves the "clean" data into a Pin—a lightweight data object hosted on Connect.
- Performance Optimization: Because the Shiny app reads from the pre-processed Pin instead of making live API calls to Veeva for every user interface (UI) element, the dashboard loads rapidly even with large clinical datasets.
- Self-Healing Workflows: Scheduled jobs can be programmed to check for the full_required flag in the API manifest. If true, the job automatically triggers a full re-download of the study data to ensure the application remains synchronized with the source system.

VEEVA API AUTHENTICATION

The technical implementation begins with authentication and the discovery of data packages. Best practices include storing credentials in Posit Connect Environment Variables rather than the code itself. Once authenticated, a unique sessionId is used for all subsequent API calls.

```
# Authenticate user session using httr2
resp <- request(paste0(url0, url1, url2)) %>%
  req_headers("Content-Type" = "application/x-www-form-urlencoded",
              "Accept" = "application/json") %>%
  req_body_form(username=user, password=pw) %>%
  req_perform()

# Extract sessionId from the JSON response
data = fromJSON(rawToChar(resp$body))
sess = data$sessionId
```

DATA PROCESSING

The R Shiny application must handle both Full Packages (generated every 24 hours) and Incremental Packages (generated every 15 minutes). The application checks the full_required boolean field; if true, it indicates a schema change and forces a full download. This ensures the app maintains a "self-healing" state relative to the source system. The R Shiny application must properly handle incremental data flow by appending new data to previously loaded data and to update existing records with changes while maintaining data integrity.

```
# Identify and download the latest available SFF package
packages <- data$data$packages %>% filter(type=="full")
mysffid <- packages[nrow(packages), "sff_id"]

resp <- request(
```

```
      paste0(
            url0,
            "/edcworkbench/api/v25.3/sff/packages/download/",
          mysffid,
            "?study_name=",
            studyid)) %>%
            req_headers("Authorization" = sess) %>%
            req_perform()
```

METADATA DRIVEN REPORTING

The Study File Format (SFF) package includes a critical manifest.json file that serves as the blueprint for the clinical data contained within the ZIP payload. This file contains all the metadata describing the clinical data, including information on data types, labels, and the structural relationship between various CSV files (for example, adverse events that are linked to concomitant medications). By parsing this manifest using a package like jsonlite, the R Shiny application can adopt a fully metadata-driven approach to dynamically generate user interface elements like data-type aware filters and new data displays tailored to the type of data (e.g. adverse events vs. vital signs). This ensures that as the clinical study design undergoes changes following protocol amendments in Veeva, the Shiny app can automatically adapt to new forms or fields without requiring manual code changes to the underlying ETL logic.

```
# Data-type aware filtering elements for Shiny UI
renderUI({
  metadata <- jsonlite::fromJSON("metadata.json")

  # Map over the JSON to create inputs
  purrr::map(1:nrow(metadata), function(i) {
    col <- metadata[i, ]

    if (col$type == "float") {
      numericInput(col$name, label = col$name, value = 0)
    } else if (col$type == "codelist") {
      selectInput(col$name, label = col$name, choices = col$options)
    } else if (col$type == "date") {
      dateRangeInput(col$name, label = col$name)
    }
  })
})
```

INTEGRATED ACTION MANAGEMENT

A critical feature of the data review tool is the ability to flag a discrepancy in the Shiny UI and have it appear in Veeva immediately. This is achieved by posting a Query Message back to the source system. To maintain GxP compliance, the tool must define the origin_sys (e.g., "Shiny_Review_Tool") and origin_user to ensure the audit trail is accurate. The benefit of this approach is that the CDB is able to maintain a complete audit trail for query actions taken both in CDB and an external data review tool, while clearly delineating which system and which user was the origin for for the query action. This provides a regulatory inspector the traceability needed to reconstruct that operational activities on the trial with 21 CFR Part 11 / Annex 11 compliant attribution. Another benefit of the query API's support for externally-integrated queries is that Veeva CDB's query metrics and operational reporting can be comprehensive for all query actions, i.e. those taken in the EDC system, the clinical workbench (CDB), and the external R Shiny data review tool. Since the raw query data is part of the SFF API payload, the data review tool can also provide additional intelligence and operational reporting related to queries, protocol deviations, and other user actions.

```r
# Post a data query back to Veeva
post_cdms_data_query <- function(study_name, id, message, session_id=NULL,
type="item_based") {
  if (is.null(session_id)) {
    session_id = get_session_id()
  }

  veeva_setup <- yaml::read_yaml(file.path("./data/veeva_setup.yaml"))

  query_creation_metadata <- dplyr::case_when(
    type == "item_based" ~ veeva_setup$queries$item_based,
    type == "event_based" ~ veeva_setup$queries$event_based,
    .default = NULL
  )

  if (is.null(query_creation_metadata)) {
    stop(paste0("Unknown query type provided: ", type, ". Supported types:
'item_based', 'event_based'."))
  }

  req_body <- list(
    "study_name" = study_name,
    "manual" = FALSE,
    "queries"= list(
      list(
        "id" = id,
        "message" = message
      )
    )
  )
  response <-  httr2::request(query_creation_metadata$endpoint) %>%
    httr2::req_method("POST") %>%
    httr2::req_headers(
      Authorization = session_id,
      Accept = query_creation_metadata$accept,
      `Content-Type` = query_creation_metadata$content_type
    ) %>%
    httr2::req_body_json(req_body) %>%
    httr2::req_perform() %>%
    httr2::resp_body_json()
  return(response)
}
```

CONCLUSION AND FUTURE OUTLOOK

This approach replaces static Excel trackers with a live, bi-directional tool, significantly reducing query cycle times and reducing the risk of documentation deficits that could arise during a regulatory inspection or audit. While the proof of concept focuses on manual queries, future iterations will explore other system actions and end user tasks, triggered from R Shiny directly back to Veeva applications:

- use of AI/ML within R to automatically suggest queries before a human reviewer sees the data
- protocol deviation identification and tracking
- clinical team observations that are annotated directly on clinical data
- RBQM metric calculations like KRIs and QTLs

REFERENCES

Veeva Clinical Data API Documentation: https://developer-cdms.veevavault.com/cdb-api/23.2/#getting-started

“Check Yourself: Edit Checks and Queries Made Easy with Shiny.” Phuse EU Connect 2025. Maya Gans & Kirsty Lauderdale. https://phuse.s3.eu-central-1.amazonaws.com/Archive/2025/Connect/EU/Hamburg/PRE_DV10.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Pavel Burmenko
Company: Veeva Systems
Email: Pavel.Burmenko@veeva.com

Author Name: Terek Peterson
Company: Atorus Research
Email: Terek.Peterson@atorusresearch.com