Data Integration Best Practices – Sept. 2025 Community Challenge

October 2025

Anaplan

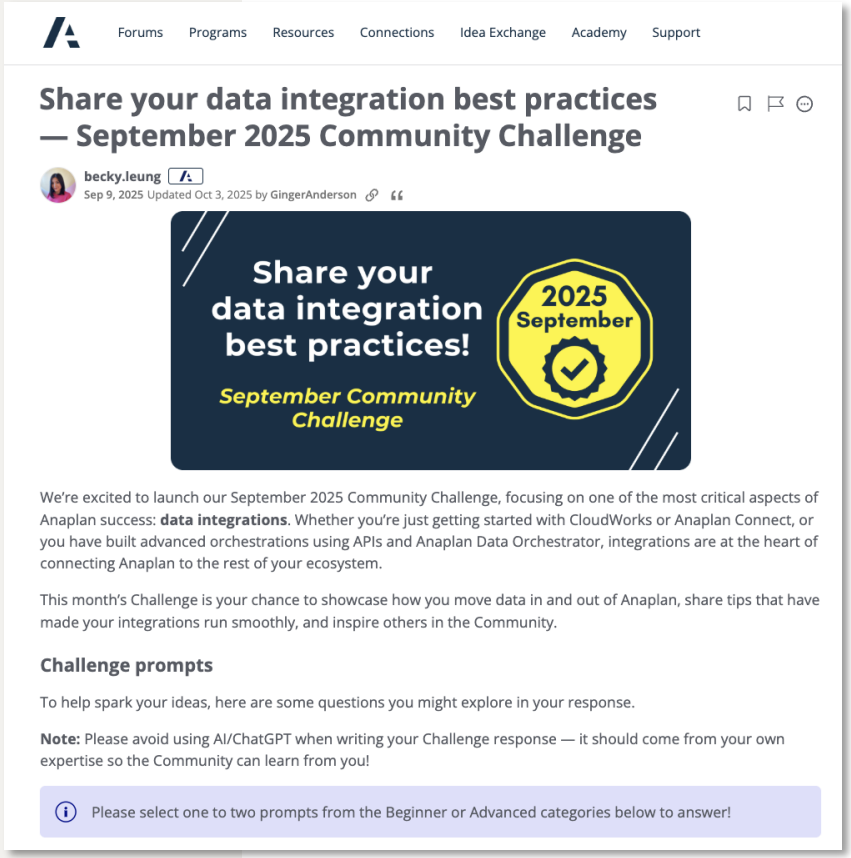
Challenge Overview

- Purpose:
  - Encourage Anaplan Community members to share integrations best practices.

- Prompts:
  - Beginner
    - File imports/exports, Saved Views, CloudWorks vs. Anaplan Connect
  - Advanced
    - ADO pipelines, APIs, Error handling, Performance tuning

Participation Snapshot

- 16 participants earned the challenge badge
- Mix of beginners and advanced users
- Tools Discussed:
  - CloudWorks
  - Anaplan Connect
  - REST APIs
  - ADO
- Common Themes:
  - Simplicity over Complexity
  - Automation
  - Reliability



Common Themes (Beginner Integration)

1. Start Simple
   - “Get used to loading the data manually(CSVs) before automating.” – andrewtye
2. Clear Naming Conventions
   - “Use detail in your saved view name… adding the destination model can save time.” – matthewreed
3. Choose the Right Tool
   - CloudWorks for ease
   - Anaplan Connect or API for power & flexibilty

Common Themes (Advanced Integration)

1. Orchestration
   - “Use ADO to set up a series of actions that run in order (e.g., import master data, then transactions.” – SunnyNafraz
2. Logging & Monitoring
   - “Always check log files after an import. Set up notifications for failures.” – SunnyNafraz
3. Import Mapping/Code
   - “When files lack a unique key… use a batch script to add a serial number column.”– Prajjwal88

Common Themes (Mistakes & Learnings)

Top Lessons:

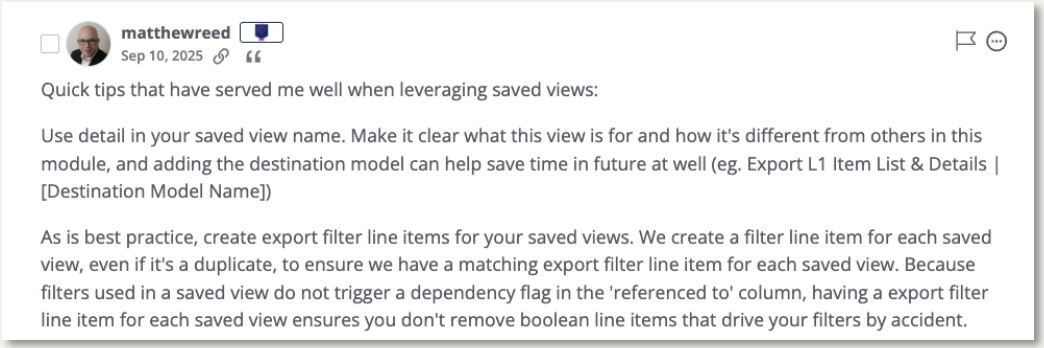
- Not checking import logs early enough
- Missing unique identifiers in transactional data
- Poor naming of saved views and filters
- Overcomplicating before validating data quality

Proper Pattern:

- Manual testing -> Automated orchestration -> Alerting & Error recovery -> Continuous Optimization

Spotlight Contributor #1 - matthewreed

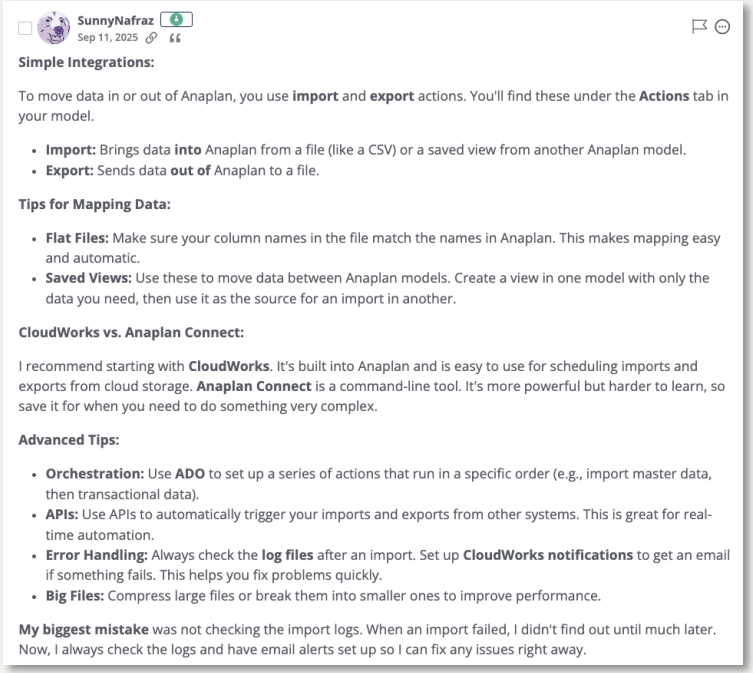
- Why Highlighted:
  - Shared foundational best practices that improve long-term model integrity.
- Key Contributions:
  - Strong naming conventions for saved views and exports.
  - Practical Tip: Use dedicated filter line items to avoid dependency issues.
- Impact:
  - Reduces accidental breakages and confusion in recurring integrations.



Spotlight Contributor #2 - SunnyNafraz

- Why Highlighted:
  - Demonstrated a complete integration lifecycle mindset.
- Key Contributions:
  - Step-by-step orchestration design using ADO.
  - Focus on logging, notifications, and continuous improvement.
  - Candid reflection: “My biggest mistake was not checking import logs early.”

- Impact:
  - Promotes a proactive, quality-driven approach to integration management.



SunnyNafraz
Sep 11, 2025

Simple Integrations:

To move data in or out of Anaplan, you use **import** and **export** actions. You'll find these under the **Actions** tab in your model.

- **Import:** Brings data **into** Anaplan from a file (like a CSV) or a saved view from another Anaplan model.
- **Export:** Sends data **out of** Anaplan to a file.

Tips for Mapping Data:

- **Flat Files:** Make sure your column names in the file match the names in Anaplan. This makes mapping easy and automatic.
- **Saved Views:** Use these to move data between Anaplan models. Create a view in one model with only the data you need, then use it as the source for an import in another.

CloudWorks vs. Anaplan Connect:

I recommend starting with **CloudWorks**. It's built into Anaplan and is easy to use for scheduling imports and exports from cloud storage. **Anaplan Connect** is a command-line tool. It's more powerful but harder to learn, so save it for when you need to do something very complex.

Advanced Tips:

- **Orchestration:** Use **ADO** to set up a series of actions that run in a specific order (e.g., import master data, then transactional data).
- **APIs:** Use APIs to automatically trigger your imports and exports from other systems. This is great for real-time automation.
- **Error Handling:** Always check the **log files** after an import. Set up **CloudWorks notifications** to get an email if something fails. This helps you fix problems quickly.
- **Big Files:** Compress large files or break them into smaller ones to improve performance.

My biggest mistake was not checking the import logs. When an import failed, I didn't find out until much later. Now, I always check the logs and have email alerts set up so I can fix any issues right away.

Key Takeaways for Practitioners

- Start with a **simple manual load** before automating
- Maintain **strong naming conventions** for saved views and filters
- Match tool choice to complexity: **CloudWorks** for simple, **ADO/API** for advanced
- **Add monitoring and error alerts early** in design
- Ensure **unique identifiers** exist in source data
- Document your integration flows and lineage maps

Next Steps & Call to Action

1. Review our current integration matrix
   - Identify gaps in monitoring, naming, or reliability.
2. Adopt quick wins:
   - Enforce naming standards.
   - Add email notifications for failures.
3. Stretch goal:
   - Convert a manual process to an automated ADO workflow.
4. Share learnings internally:
   - Present these insights at the next COE or community of practice meeting.

Q&A

Thank you!