

Use Cloud Platform Integration Suite to extract Ariba Data into SAP Datasphere

Todd Hanna, SAP
October 11, 2023

Public

Agenda

Background – Ariba Analytical API

- Current Challenge – what are we solving
- Understanding the Ariba Analytical API
- Example Postman Scripts to pull Data

Cloud Platform Integration Suite & SAP Datasphere - Solution Overview

- High Level solution overview
 - SAP Components and Architecture
- Details on the Integration Flows and Mapping Files
- Setup and Configuration

Available Integrations

- Datasphere Business Content: Ariba Spend Analysis
- SAC Business Content: Ariba Procure to Order

Customization opportunity

- Steps to repurpose Integration Flows against other source views

Background - Ariba Analytical API

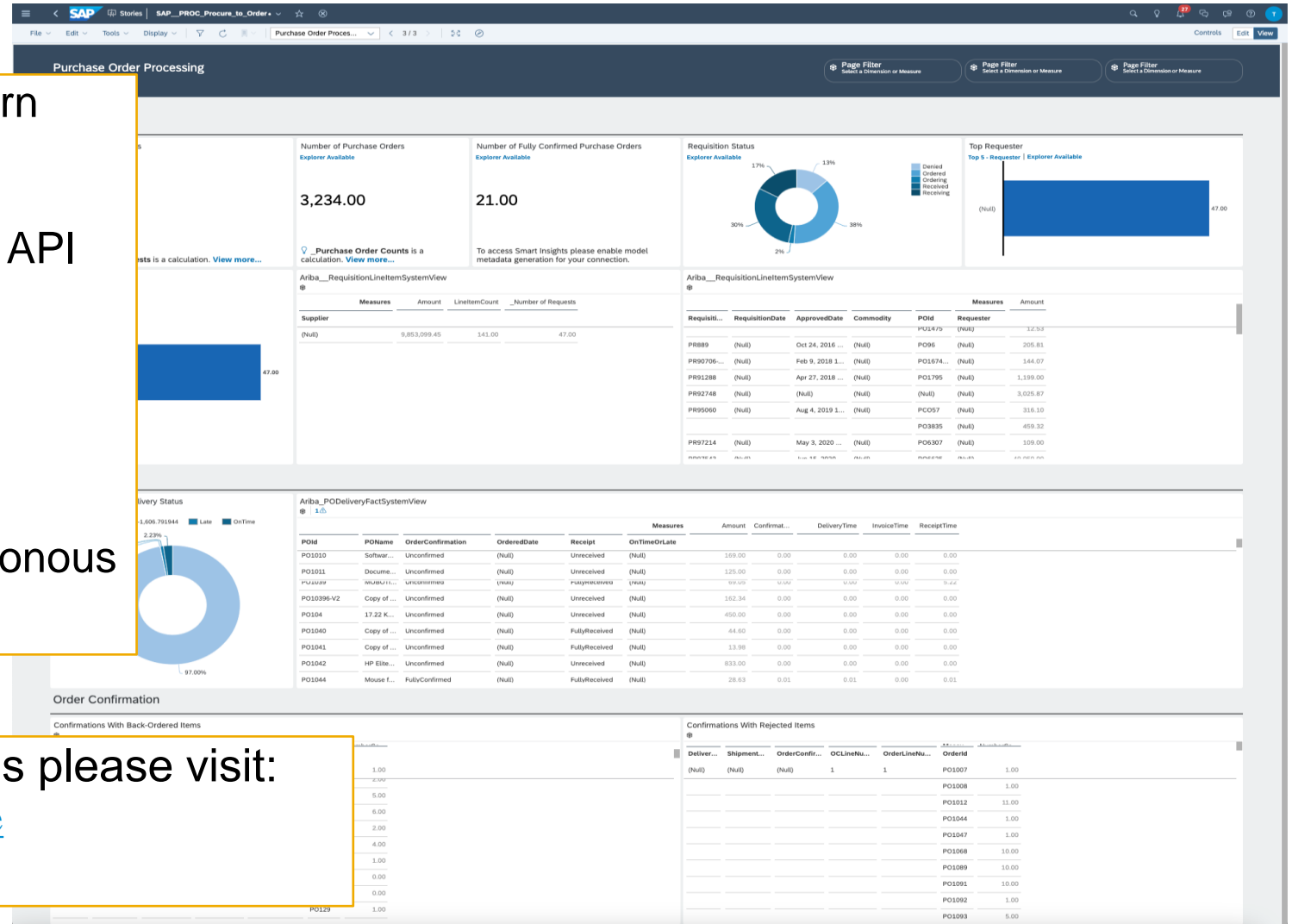
Current Challenge

How to get data from Ariba into modern Analytical tool such as SAC?

- Ariba Data available via Analytical API
- Analytical API - two variants:
 - Synchronous (operational)
 - Asynchronous (analytical)
- This solution focusses on Asynchronous due to its high volume data flow

For more information on Ariba's APIs please visit:

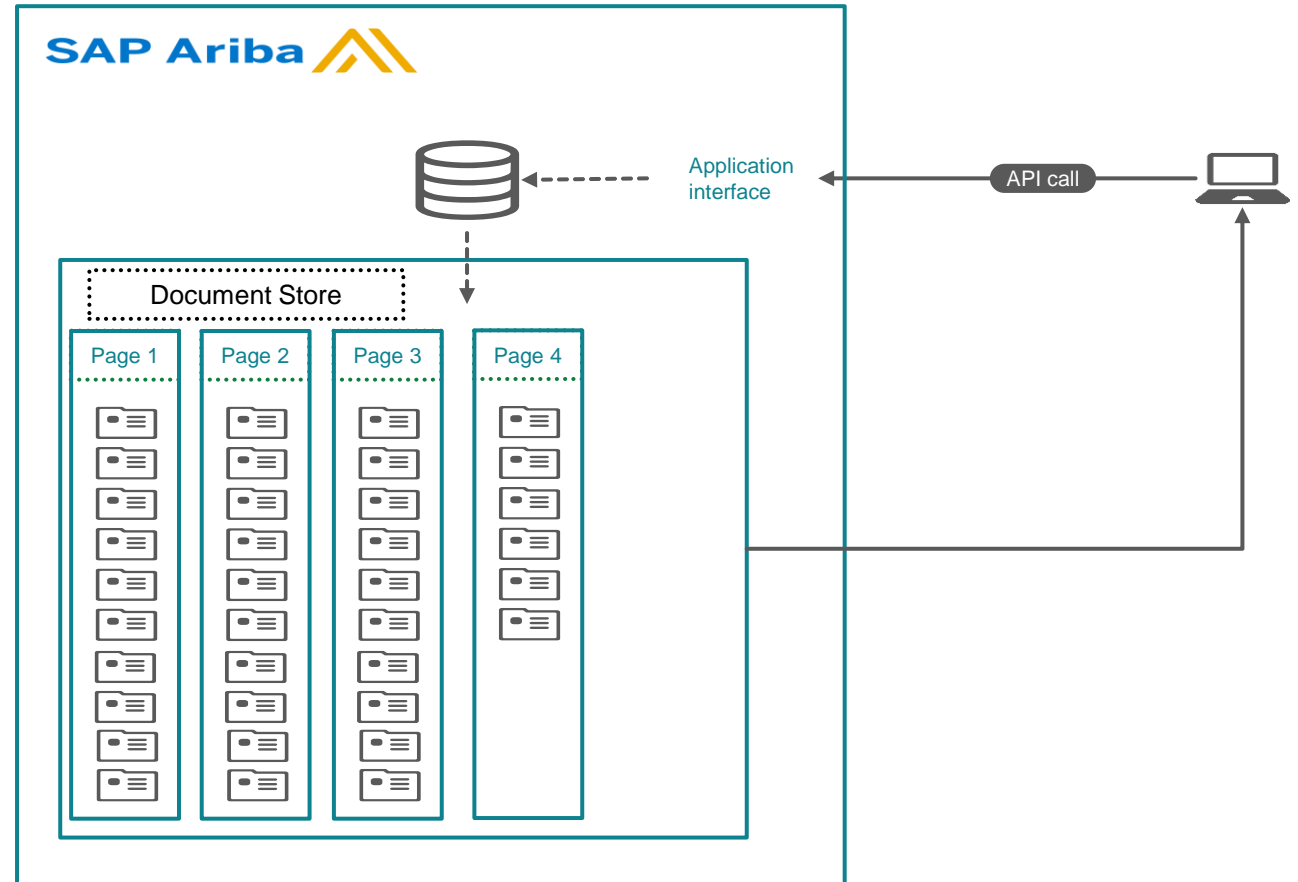
- <https://developer.ariba.com/api/welcome>
- <https://developer.ariba.com/api/guides>



Asynchronous API Response

Asynchronous API

- Designed for higher volume Analytical use cases
- ~ 150 System Views (facts and dimensions) accessible via this API
- API response is in .ZIP format
- Single ZIP is up to 50K rows
- If response is > 50K – Multiple ZIPs generated
- If response is large enough, multiple PAGES created (up to 10 Zips / Page)
- Pages and Zips have unique IDs



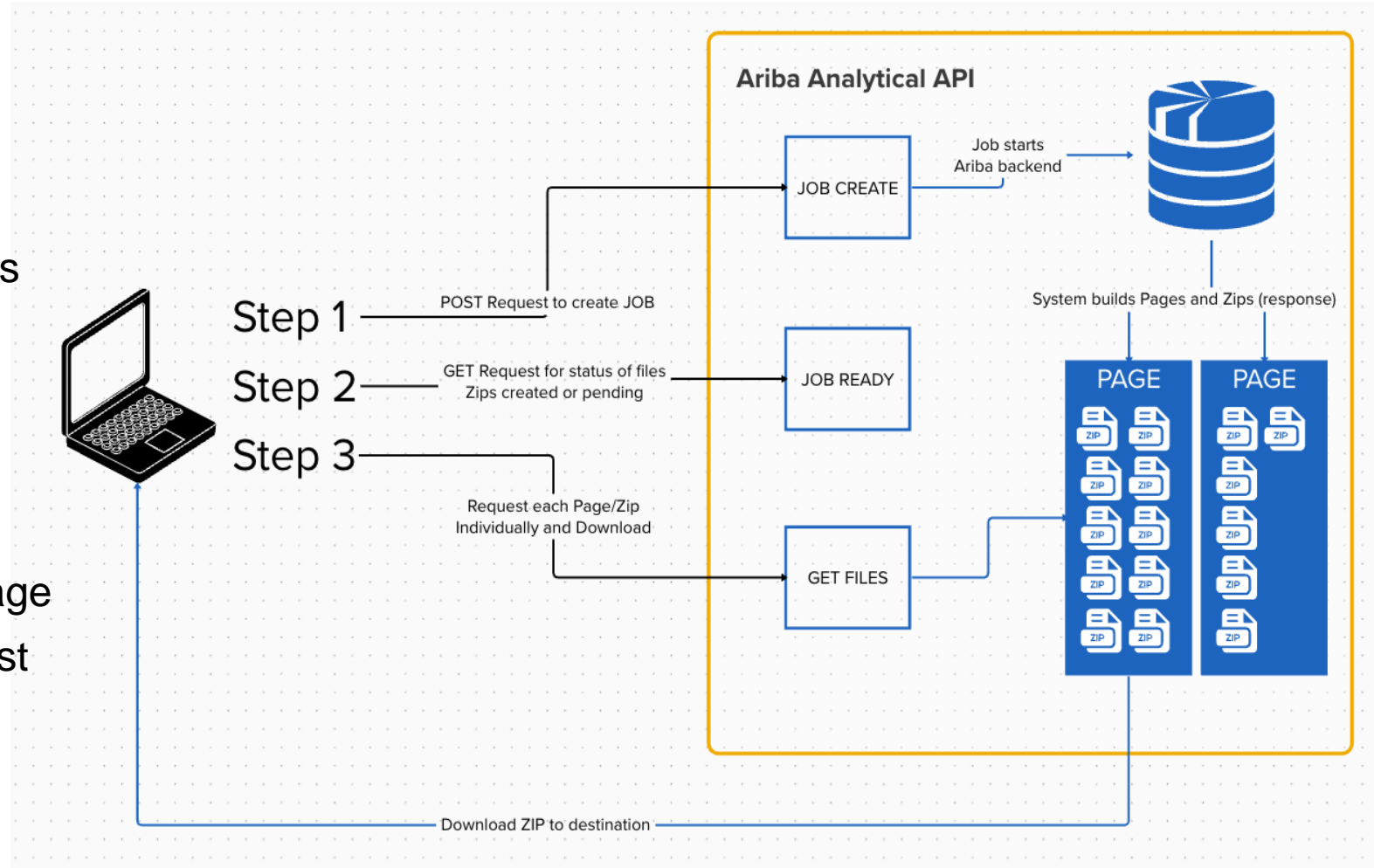
Example – Request for Invoices returns 1.8M records

- Data is batched into 4 pages
- Pages 1-3 contain 500K records in 10 zips
- Page 4 contains 300K records in 6 zips

Working with the Asynchronous API

Steps to Retrieve System Views

- POST request
 - Creates job on Ariba Backend
 - Job can create multiple Pages/Zips
- GET request
 - Check status of Job
- GET request
 - Download ZIP file from specific Page
 - Each ZIP requires separate request

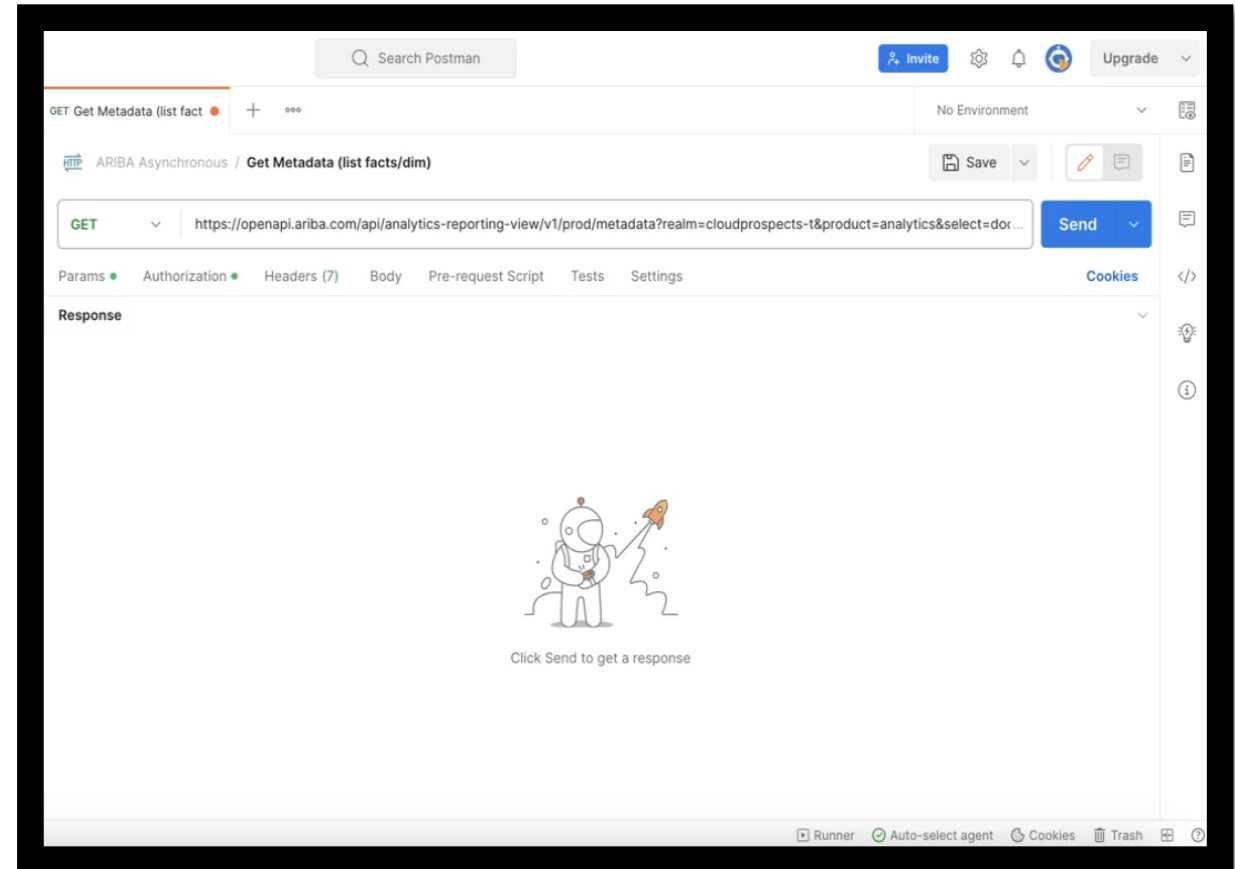


GET list of System Views

https://openapi.ariba.com/api/analytics-reporting-view/v1/prod/metadata?realm=<REALM_NAME>&product=analytics&select=documentType

API GET Request for list of System Views

- Replace <REALM_NAME> with your Ariba Realm
- JSON Response with list of Facts and Dimensions
- Used in future calls to get structure of individual views
- Add 'SystemView' to suffix of any Fact or Dimension name

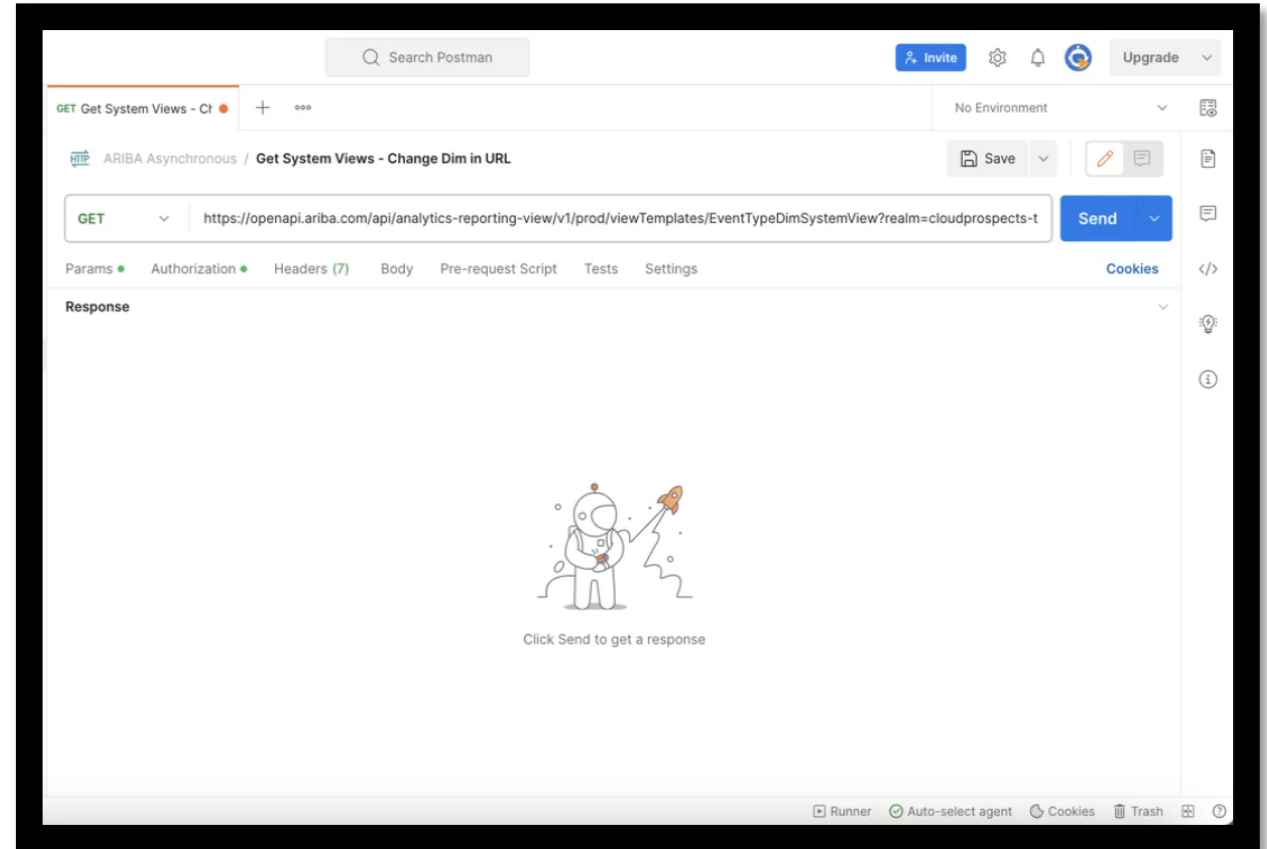


GET structure for a specific System Table

https://openapi.ariba.com/api/analytics-reporting-view/v1/prod/viewTemplates/EventTypeDimSystemView?realm=<REALM_NAME>

API GET Request to get structure (columns) for a specific table

- Replace <REALM_NAME> with your Ariba Realm
- Highlighted System View EventTypeDim is target – add SystemView to any Fact or Dim from previous response
- JSON Response contains column names and filtering params

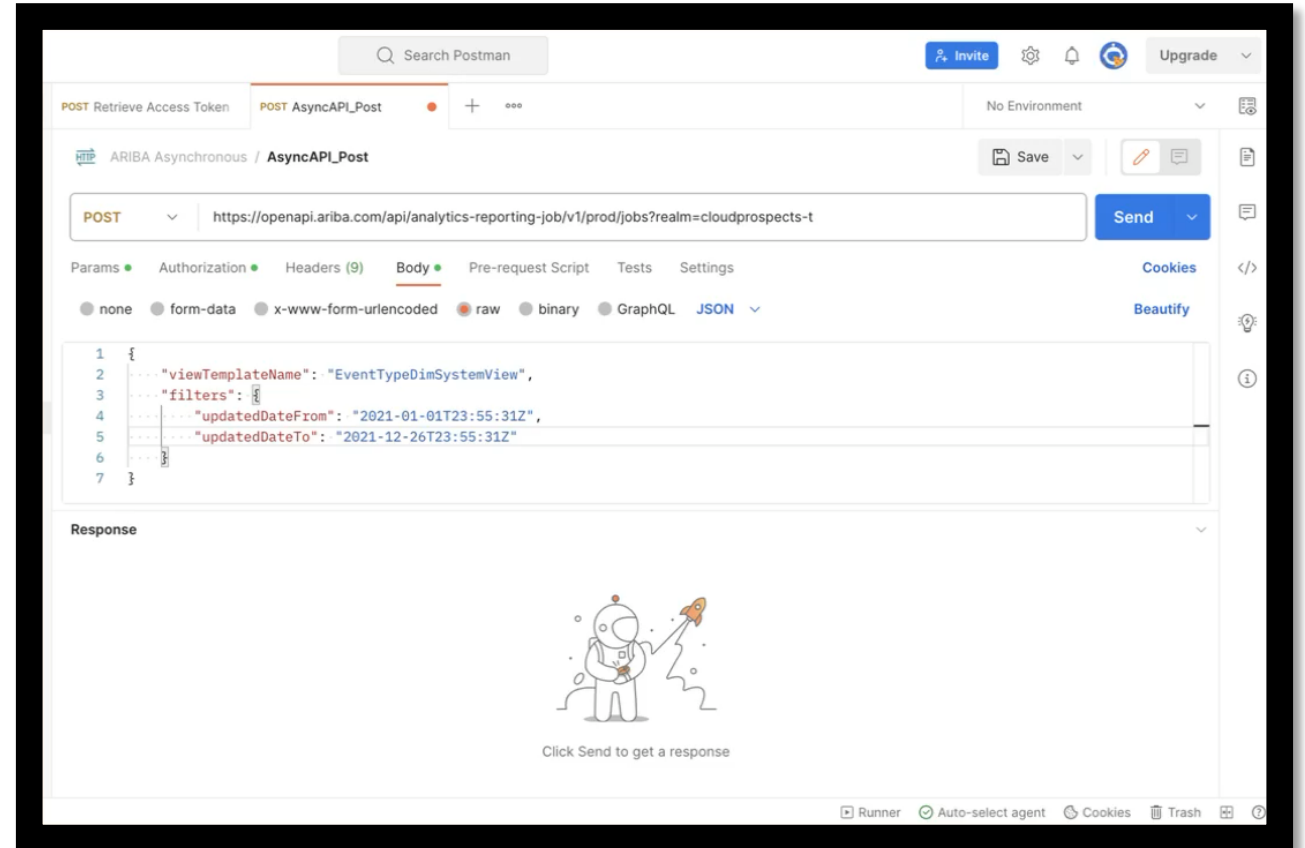


POST Job to build ZIP files for a System View

https://openapi.ariba.com/api/analytics-reporting-job/v1/prod/jobs?realm=REALM_NAME

API POST Request to have Ariba build out a response containing data (ZIP files)

- Replace <REALM_NAME> with your Ariba Realm
- Need to provide info in Body of request for
 - System View requested
 - Dataset filters
- Request kicks off Job to build response payload "Asynchronously"

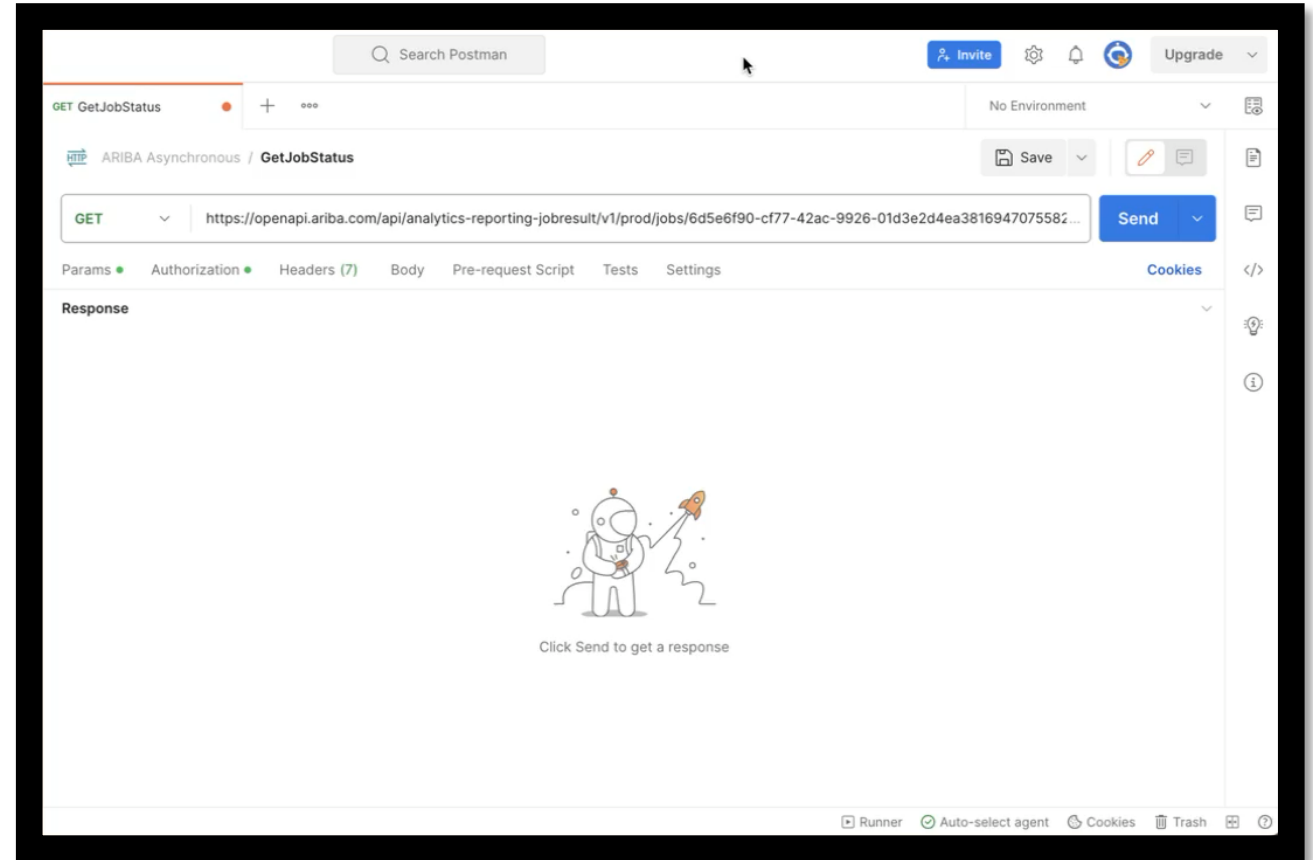


GET status of Ariba Job

https://openapi.ariba.com/api/analytics-reporting-jobresult/v1/prod/jobs/6d5e6f90-cf77-42ac-9926-01d3e2d4ea381694707558291?realm=REALM_NAME>

API GET Request to check status of JOB

- Replace <REALM_NAME> with your Ariba Realm
- Highlighted JOB ID from response of previous Request
- If Job not ready: response = “Pending”
- If Job ready: Completed + ZIP file name(s)

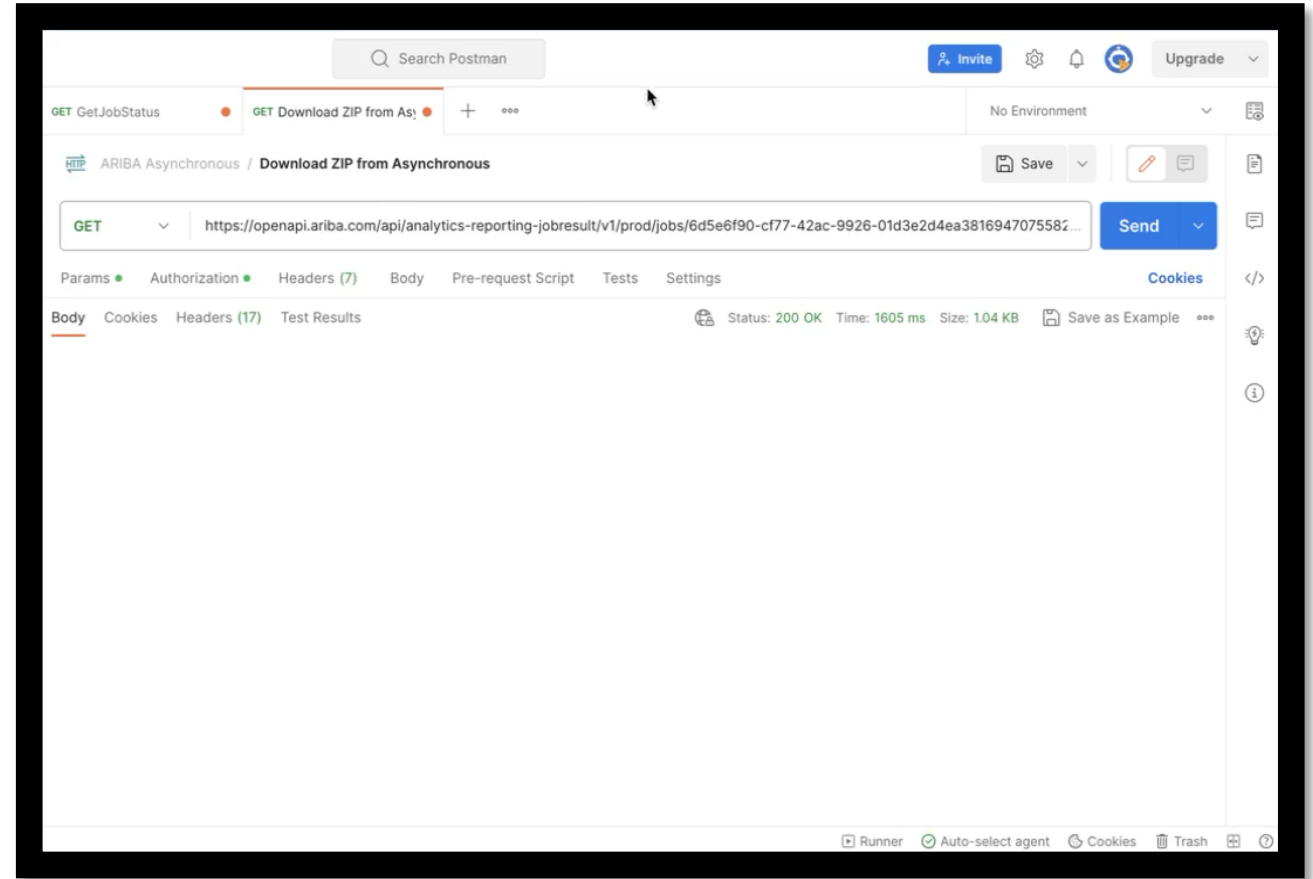


GET (download) Ariba Job payload

https://openapi.ariba.com/api/analytics-reporting-jobresult/v1/prod/jobs/6d5e6f90-cf77-42ac-9926-01d3e2d4ea381694707558291/files/Flmjd4gwj.zip?realm=<REALM_NAME>

API GET Request for the data from the Ariba Job

- Replace <REALM_NAME> with your Ariba Realm
- Highlighted JOB ID from response of previous Request
- Highlighted ZIP name from response of previous Request
- Download the Zip file
- ZIP file contains records in JSON format

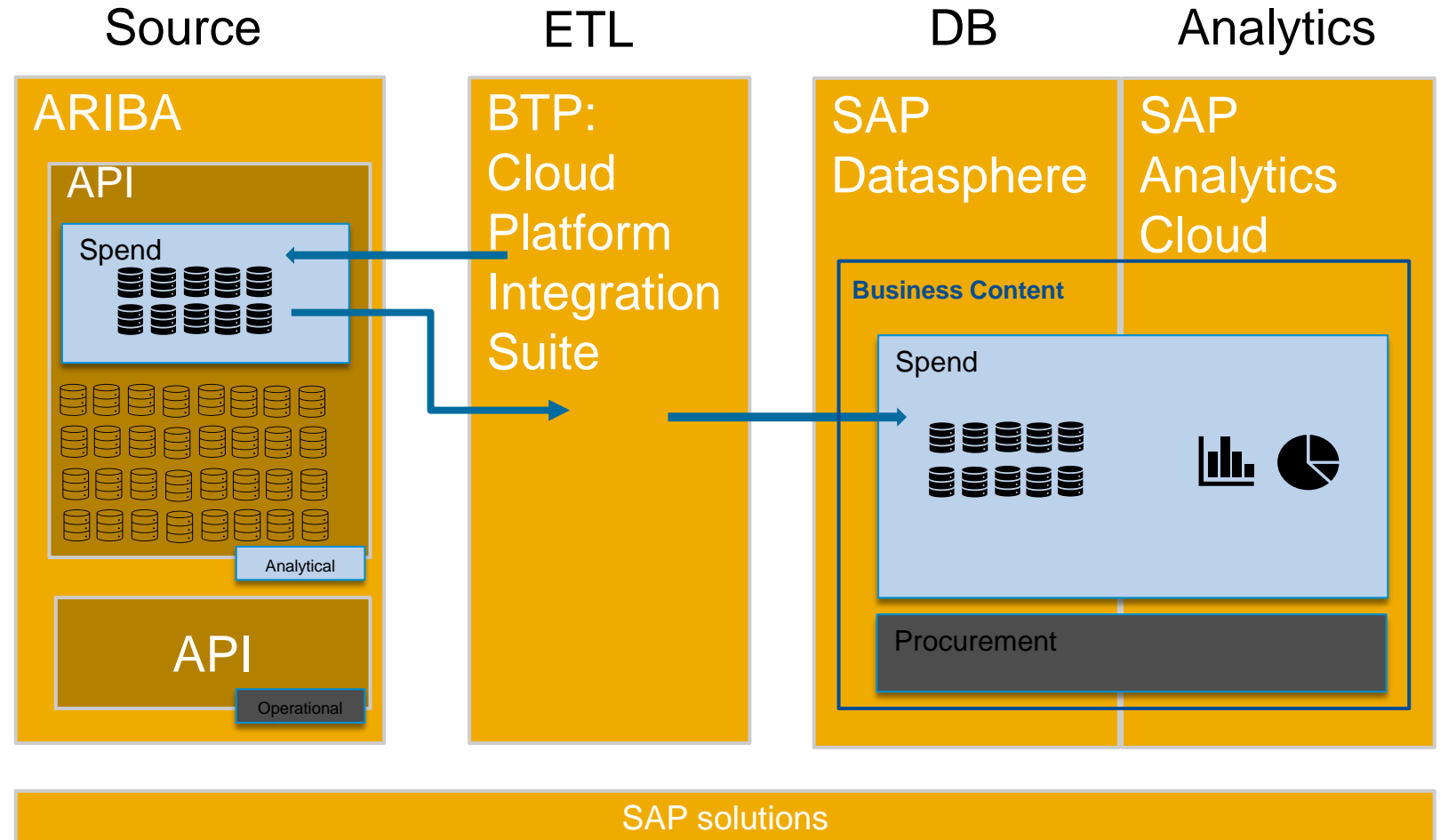


Cloud Platform Integration Suite & SAP Datasphere – Solution Overview

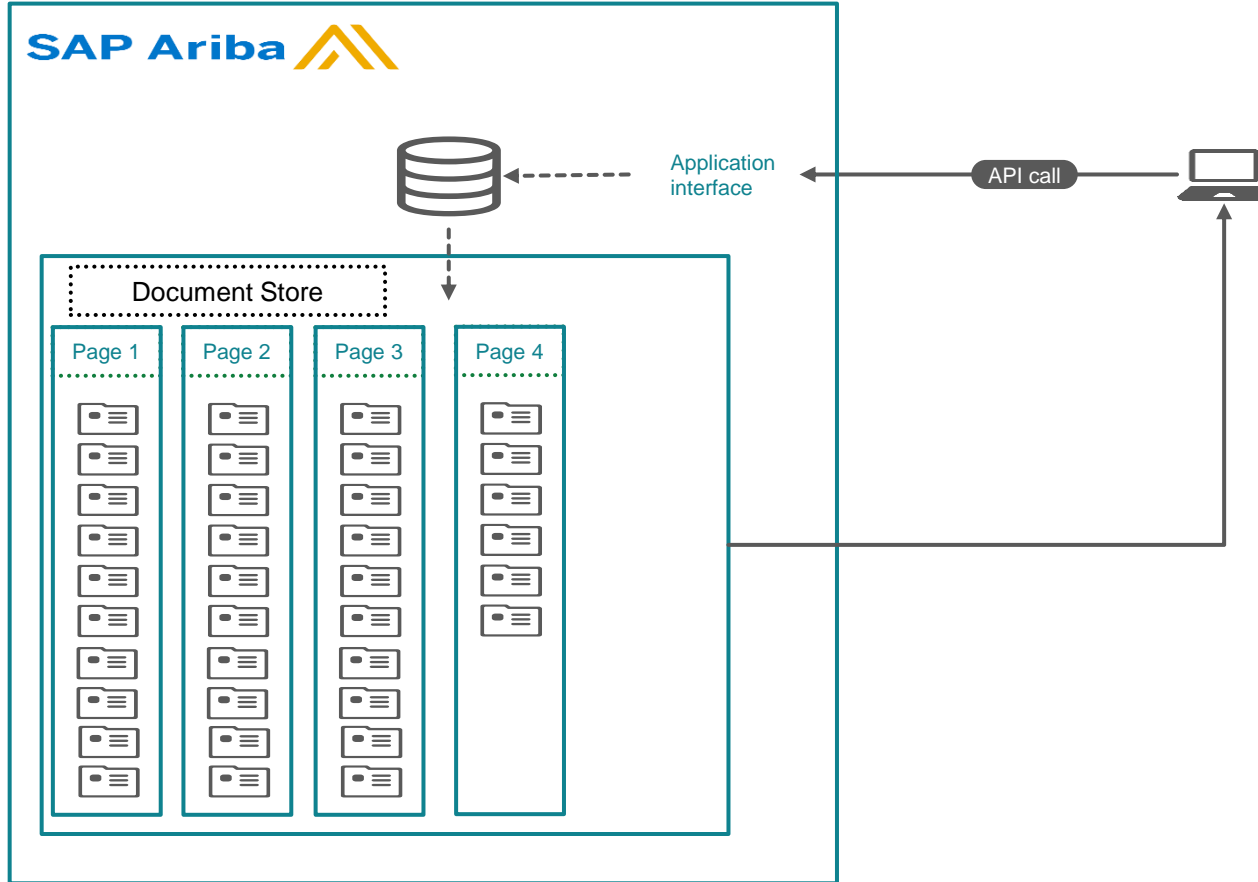
SAP Datasphere Business Content

Started with Spend Analysis Business Content

- 10 Ariba Source Views
- Downloadable SAC and Datasphere Content (views and stories)
- Need mechanism to ETL Ariba Data to underlying Tables
- Wanted SAP solution



SAP Cloud Platform Integration Suite - ETL Layer

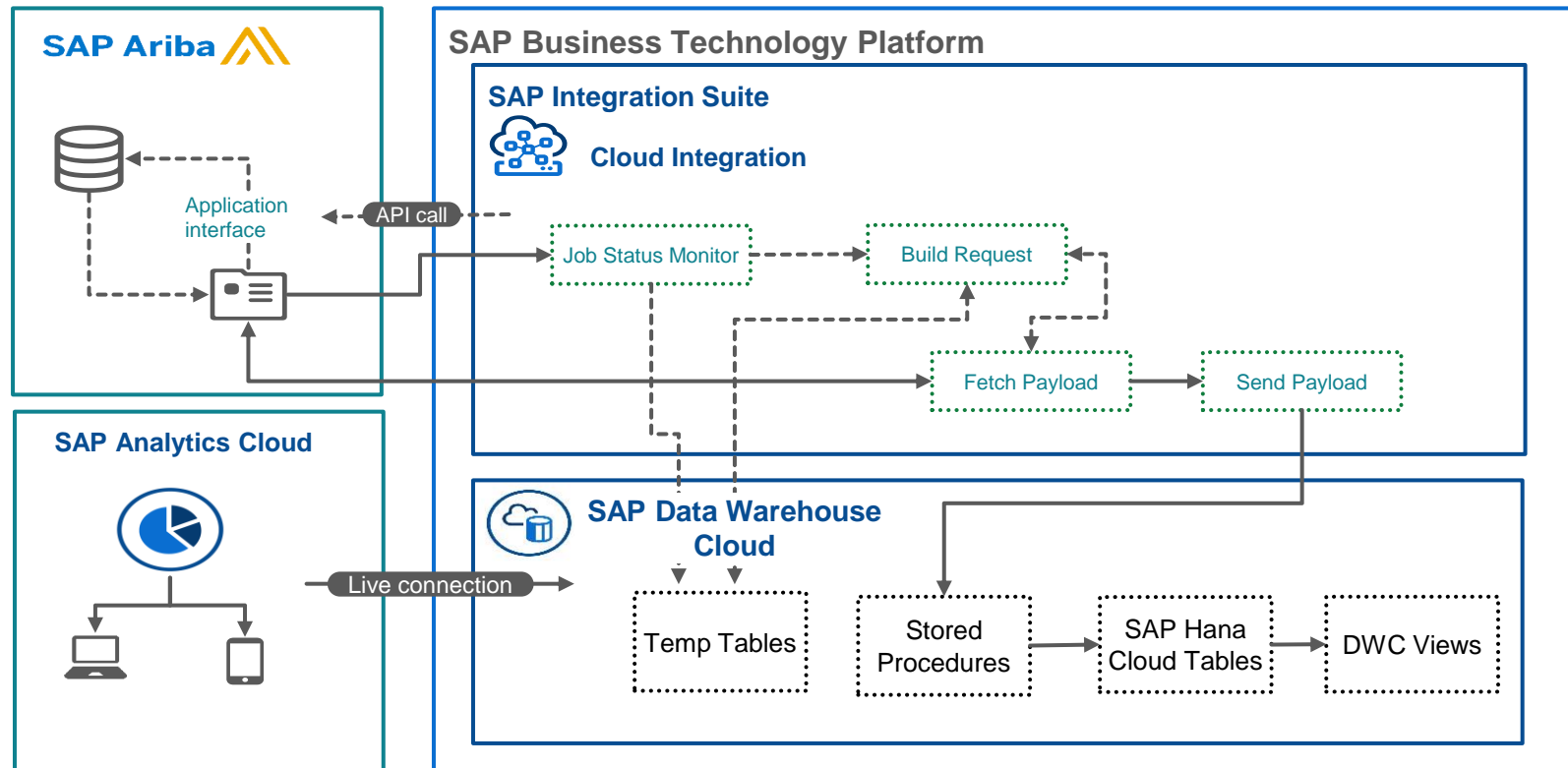


1. Submit a Job request via Ariba API
2. Ariba builds result set
3. Result Set is batched into 1 or many Pages containing up to 10 zips
4. Each Zip contains up to 50K records (zipped JSON)
5. API response returns Page IDs and Zip Names
6. Separate request to retrieve a Zip

Example – Request for Invoices returns 1.8M records

- Data is batched into 4 pages
- Pages 1-3 contain 500K records in 10 zips
- Page 4 contains 300K records in 6 zips

Integration Flows - Architecture



1. Establish Connection between Ariba/CPIS/Datasphere
2. CPIS to submit view template request to Ariba
3. CPIS checks if Pagination is needed
4. CPIS retrieves the information from Ariba.
5. CPIS converts the data and writes to Datasphere
6. SAC to produce reports based on Data written to Datasphere.

Run times*

- Dimension Tables(8)
 - For customers with about 100,000 accounts, 200,000 cost centers, 50,000 Parts etc, the loading of the data into DWC from Ariba would take about 2 hours of processing time.
- Fact tables(2)
 - Fact tables would contain millions of records, 1M records take about 1 hour of processing time.
 - For fact tables that exceed more than 1M records in a run, it is advised to run the data load in smaller chunks i.e Monthly/Weekly/Daily.

Integration Suite – Flows & Mappings

Integration Suite Flow Package contains:

- 6 Integration Flows
- 1 mapping for every Ariba System View being pulled

Spend Analysis uses 10 System Views – 10 mapping files

The screenshot displays the SAP Integration Suite web interface. The left sidebar shows the navigation menu with 'Integrations' selected. The main content area shows the 'Ariba Spend Analysis' package details, including a description and version information. Below this, a table lists the package's artifacts, categorized into 'Mapping Files' and 'Integration Flows'.

Name	Type	Version	Actions
<input type="checkbox"/> Ariba_AccountDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_CompanySiteDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_ContractDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_CostCenterDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_InvoiceLineItemFactMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_PartDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_POLineItemFactMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_SourceSystemDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_SupplierDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> Ariba_UNSPSCDimMap Created	Message Mapping	1.0.0	[Link]
<input type="checkbox"/> AribaSpend_Step_0 Trigger Main ISM iFlow Created	Integration Flow	1.0.3	[Link]
<input type="checkbox"/> AribaSpend_Step_1 Intelligent Spend management - Main iFlow Created	Integration Flow	2.1	[Link]
<input type="checkbox"/> AribaSpend_Step_2 Intelligent Spend management - Job Status Checker Created	Integration Flow	2.1	[Link]
<input type="checkbox"/> AribaSpend_Step_3 Intelligent Spend management - Fetch data - File by File Created	Integration Flow	2.1	[Link]
<input type="checkbox"/> AribaSpend_Step_4 Intelligent Spend management - Fetch Redirected FQDN Created	Integration Flow	2.1	[Link]
<input type="checkbox"/> AribaSpend_Step_5 Intelligent Spend management - Final loading process Created	Integration Flow	2.1	[Link]

Integration Flow 1: 'Step0'

Step 0:

- Kicks off process
- Submits Job to Ariba API
- Uses parameters from 'Step1'

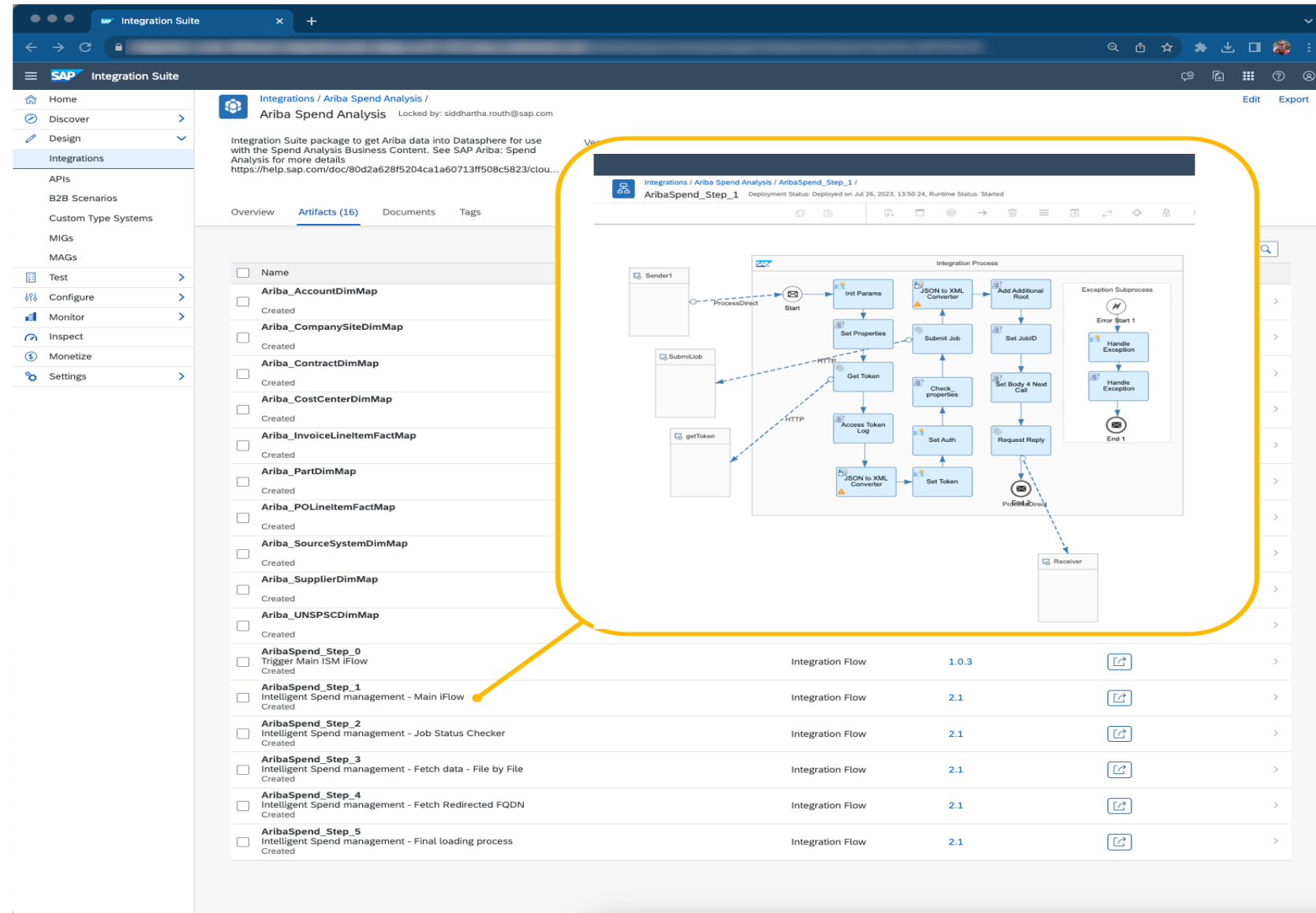
The screenshot displays the SAP Integration Suite interface. On the left, a sidebar menu shows navigation options: Home, Discover, Design, Integrations (selected), APIs, B2B Scenarios, Custom Type Systems, MIGs, and MAGs. The main content area is titled 'Integrations / Ariba Spend Analysis / Ariba Spend Analysis'. It includes a description of the integration suite package, a vendor (SAP), and a mode (Editable). Below this, there are tabs for Overview, Artifacts (16), Documents, and Tags. The Artifacts tab is active, showing a list of artifacts. A yellow arrow points from the 'AribaSpend_Step_0' artifact in the list to a detailed view of the integration flow diagram. The diagram, titled 'Integration Process', shows a flow starting with a 'Start Timer' event, followed by a 'Request Reply' process, and ending with an 'End' event. A 'ProcessDirect' connector links the 'Request Reply' process to a 'Trigger_L...' event. Below the diagram, there is a table listing integration flows.

Integration Flow	Version	Actions
AribaSpend_Step_0	2.1	[Edit] [Copy] [Delete] [Share]
AribaSpend_Step_1	2.1	[Edit] [Copy] [Delete] [Share]
AribaSpend_Step_2	2.1	[Edit] [Copy] [Delete] [Share]
AribaSpend_Step_3	2.1	[Edit] [Copy] [Delete] [Share]
AribaSpend_Step_4	2.1	[Edit] [Copy] [Delete] [Share]
AribaSpend_Step_5	2.1	[Edit] [Copy] [Delete] [Share]

Integration Flow 2: 'Step 1'

Step 1:

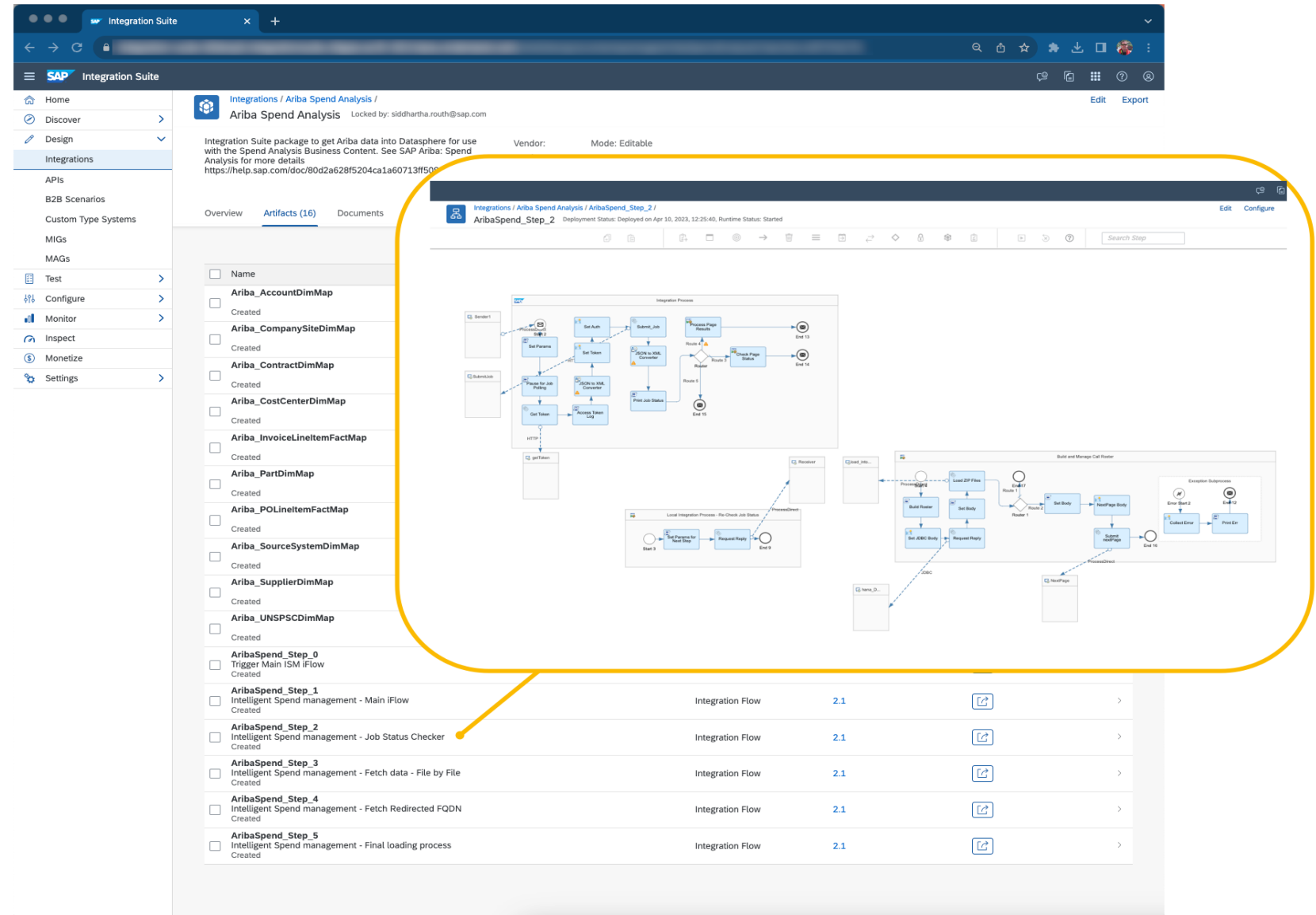
- Worker Flow for 'Step 2'
- Gathers ID of Page and Zip
- Used for multi-page and multi-zip scenario



Integration Flow 3: 'Step 2'

Step 2:

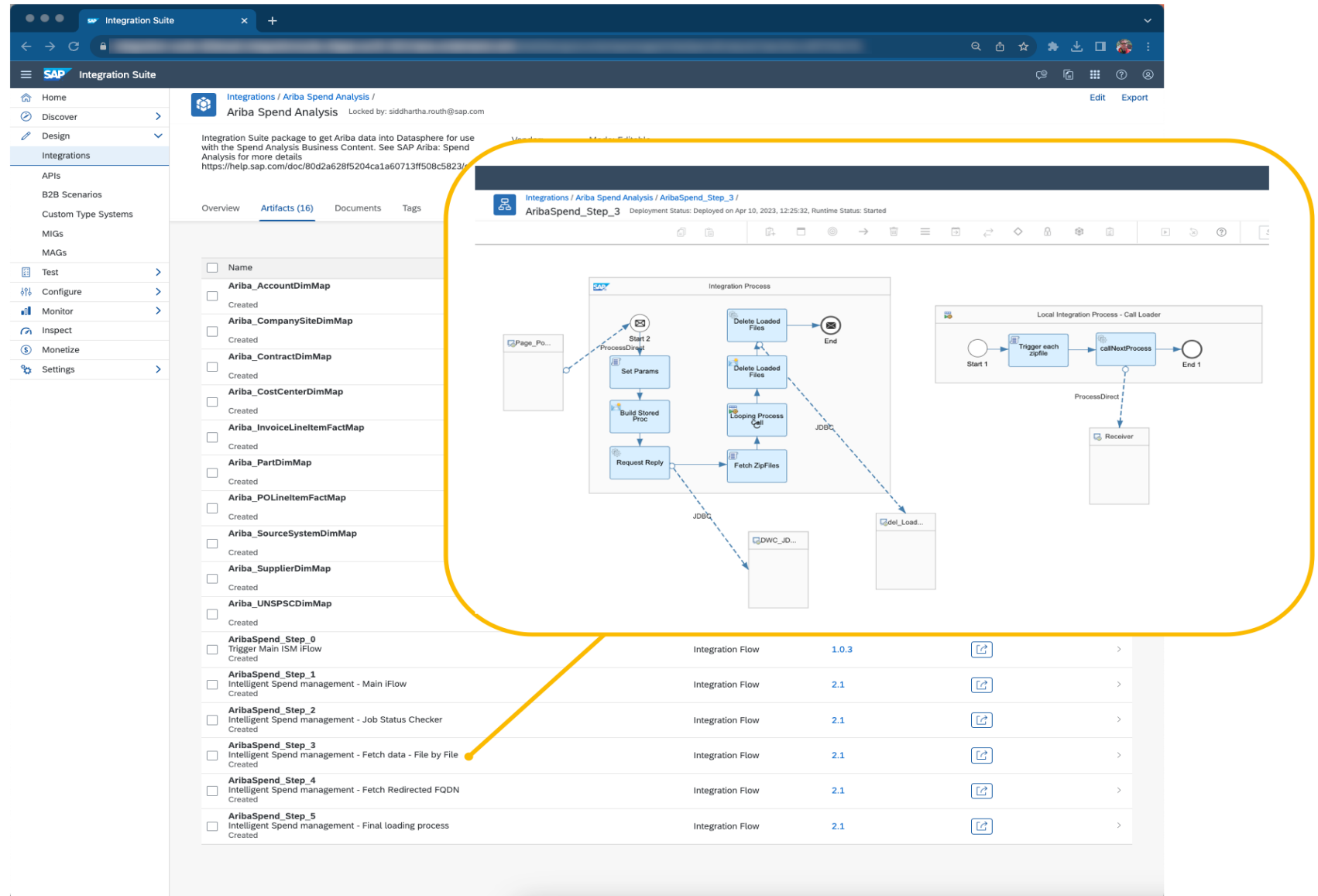
- Polls API to determine if job is 'complete'
- When job complete, collates a list of all Page / Zip combinations
 - Uses Step 1 to iterate
- When full list – triggers Step 3



Integration Flow 4: 'Step 3'

Step 3:

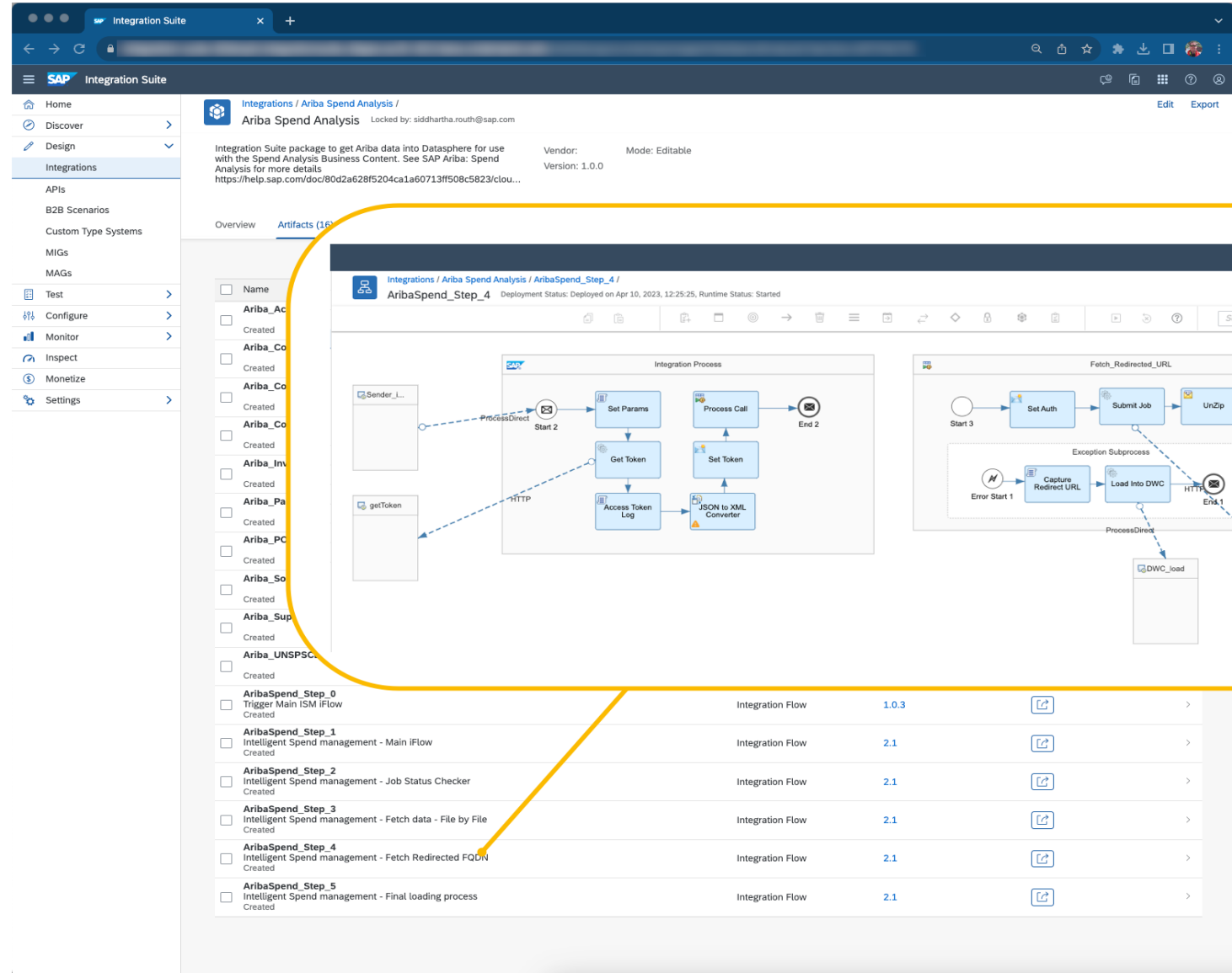
- Receives 'Page/Zip' list from Step 2
- Collates with JobID to create URL
- Passes full list of JobID/Page/ZIP to Step 4



Integration Flow 5: 'Step 4'

Step 4:

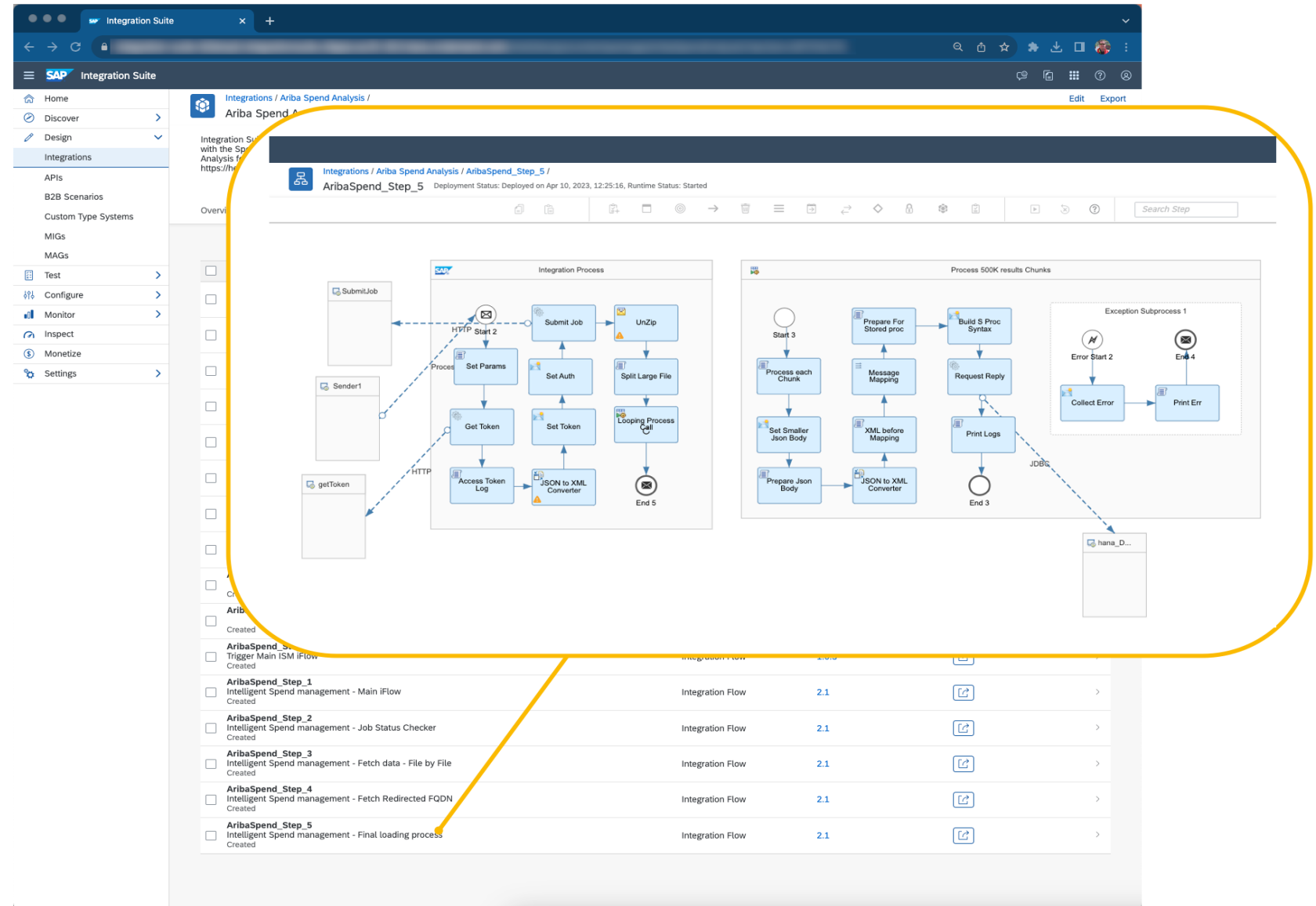
- Builds GET request from info passed in Step 3
- Follows any URL redirects (Ariba side)
- Exception Handling
- Passes GET request to Step 5



Integration Flow 6: 'Step 5'

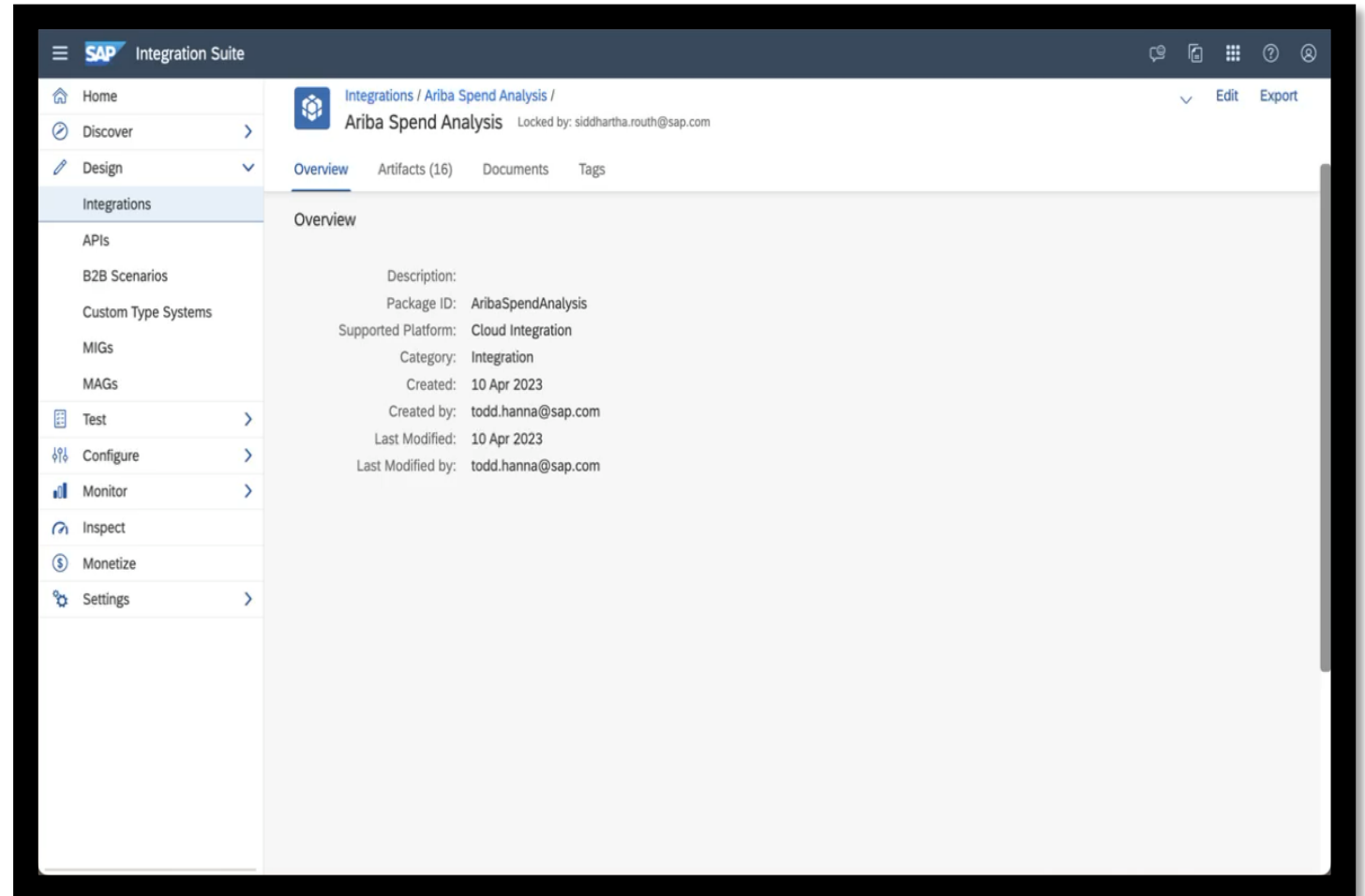
Step 5:

- Sends GET request to Ariba; Retrieves ZIP
- Unzips JSON, converts to XML; Sends XML to Datasphere
- Repeats through all ZIPs



Message Mappings

- Mechanism to map Source Views (Ariba) to Target Tables (Datasphere)
- 1 Map File per Source / Target
- Can map source to target whether or not column names match (GUI)
- Unmapped target columns populate NULLs
- Spend package: 10 Maps (1 per source System View)



Message Mappings - XSD

- Definition files are XSD: XML Schema Definition
- Source and Target each have slightly different version
- Can look at format for each in Interface
- Useful as guidance for any new tables (customization)

The screenshot displays the SAP Integration Suite Message Mapping interface. The left sidebar shows the navigation menu with options like Home, Discover, Design, Integrations, APIs, B2B Scenarios, Custom Type Systems, MIGs, MAGs, Test, Configure, Monitor, Inspect, Monetize, and Settings. The main area is titled 'Integrations / Ariba Spend Analysis / Ariba_POLineItemFactMap / Ariba_POLineItemFactMap'. It shows a message mapping between a source structure and a target structure. The source structure, 'Ariba_POLineItemFactMap', has fields: ExtraPOLineKey, ExtraPOKey, SplitAccountingNumber, ServiceStartDate, ServiceEndDate, RequiresServiceEntry, Description, StatusString, and PriceVarCost. The target structure, 'resultset', has fields: flex_string3, flex_string2, flex_string10, flex_string1, extra_pokey, extra_pokey, description, contract_id, and split_accounting_number. A mapping expression is shown at the bottom: 'Description' maps to 'description'.

Name	Type
Schen	
pol	XSD
PO	XSD

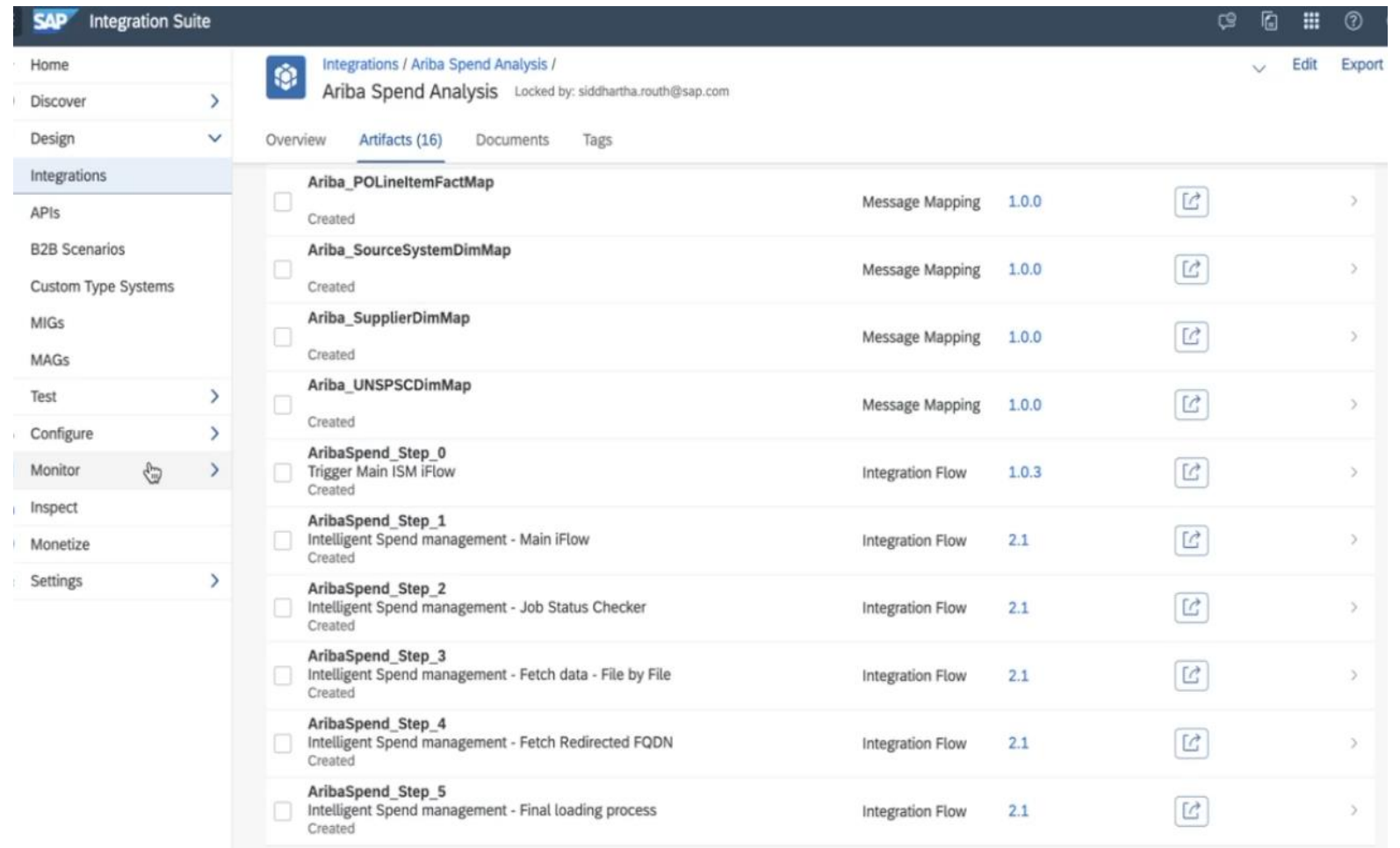
Field	Cardinality
ExtraPOLineKey	1..1
ExtraPOKey	1..1
SplitAccountingNumber	1..1
ServiceStartDate	1..1
ServiceEndDate	1..1
RequiresServiceEntry	1..1
Description	1..1
StatusString	1..1
PriceVarCost	1..1

Field	Cardinality
flex_string3	0..1
flex_string2	0..1
flex_string10	0..1
flex_string1	0..1
extra_pokey	0..1
description	0..1
contract_id	0..1
split_accounting_number	0..1

Mapping Expression: Description → description

Integration Suite – Ariba Connectivity

- Ariba connection



The screenshot displays the SAP Integration Suite interface for Ariba Spend Analysis. The left sidebar contains navigation options: Home, Discover, Design, Integrations (selected), APIs, B2B Scenarios, Custom Type Systems, MIGs, MAGs, Test, Configure, Monitor (highlighted with a mouse cursor), Inspect, Monetize, and Settings. The main content area shows the 'Artifacts (16)' tab, listing various integration artifacts. Each artifact includes a checkbox, name, description, type, version, and action icons (edit, delete, share, and expand).

Artifact Name	Description	Type	Version	Action
<input type="checkbox"/> Ariba_POLineItemFactMap	Created	Message Mapping	1.0.0	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> Ariba_SourceSystemDimMap	Created	Message Mapping	1.0.0	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> Ariba_SupplierDimMap	Created	Message Mapping	1.0.0	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> Ariba_UNSPSCDimMap	Created	Message Mapping	1.0.0	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_0	Trigger Main ISM iFlow Created	Integration Flow	1.0.3	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_1	Intelligent Spend management - Main iFlow Created	Integration Flow	2.1	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_2	Intelligent Spend management - Job Status Checker Created	Integration Flow	2.1	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_3	Intelligent Spend management - Fetch data - File by File Created	Integration Flow	2.1	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_4	Intelligent Spend management - Fetch Redirected FQDN Created	Integration Flow	2.1	[Edit] [Delete] [Share] [Expand]
<input type="checkbox"/> AribaSpend_Step_5	Intelligent Spend management - Final loading process Created	Integration Flow	2.1	[Edit] [Delete] [Share] [Expand]

Integration Suite – Datasphere Connectivity

Datasphere Space Management ->
Database User

JDBC URL:

- 'JDBC://' + HostName + Port

User:

- Database User Name

Password:

- Password

The screenshot displays the SAP Datasphere interface with two overlapping dialog boxes. The 'Edit JDBC Data Source' dialog on the left contains the following fields: Name (DWC_HANACLOUD), Description (CEG DWC test tenant), Database Type (SAP HANA Cloud), User (SAP_CONTENT#ARIBA_SPEND_ANALYSIS), Password, Repeat Password, and JDBC URL (jdbc:sap://). The 'Database User Details' dialog on the right contains: Database User Name (SAP_CONTENT#ARIBA_SPEND_ANALYSIS), Open SQL Schema (SAP_CONTENT#ARIBA_SPEND_ANALYSIS), Space Schema (SAP_CONTENT), Host Name, Port (443), Password, and a 'Request New Password' button. Yellow arrows indicate the following mappings: from 'User' in the first dialog to 'Database User Name' in the second; from 'Password' in the first dialog to 'Password' in the second; from 'JDBC URL' in the first dialog to 'Host Name' and 'Port' in the second; and from 'Database User Name' in the second dialog to 'Open SQL Schema' in the second dialog.

SAP Datasphere HANA Layer (SQL Scripts)

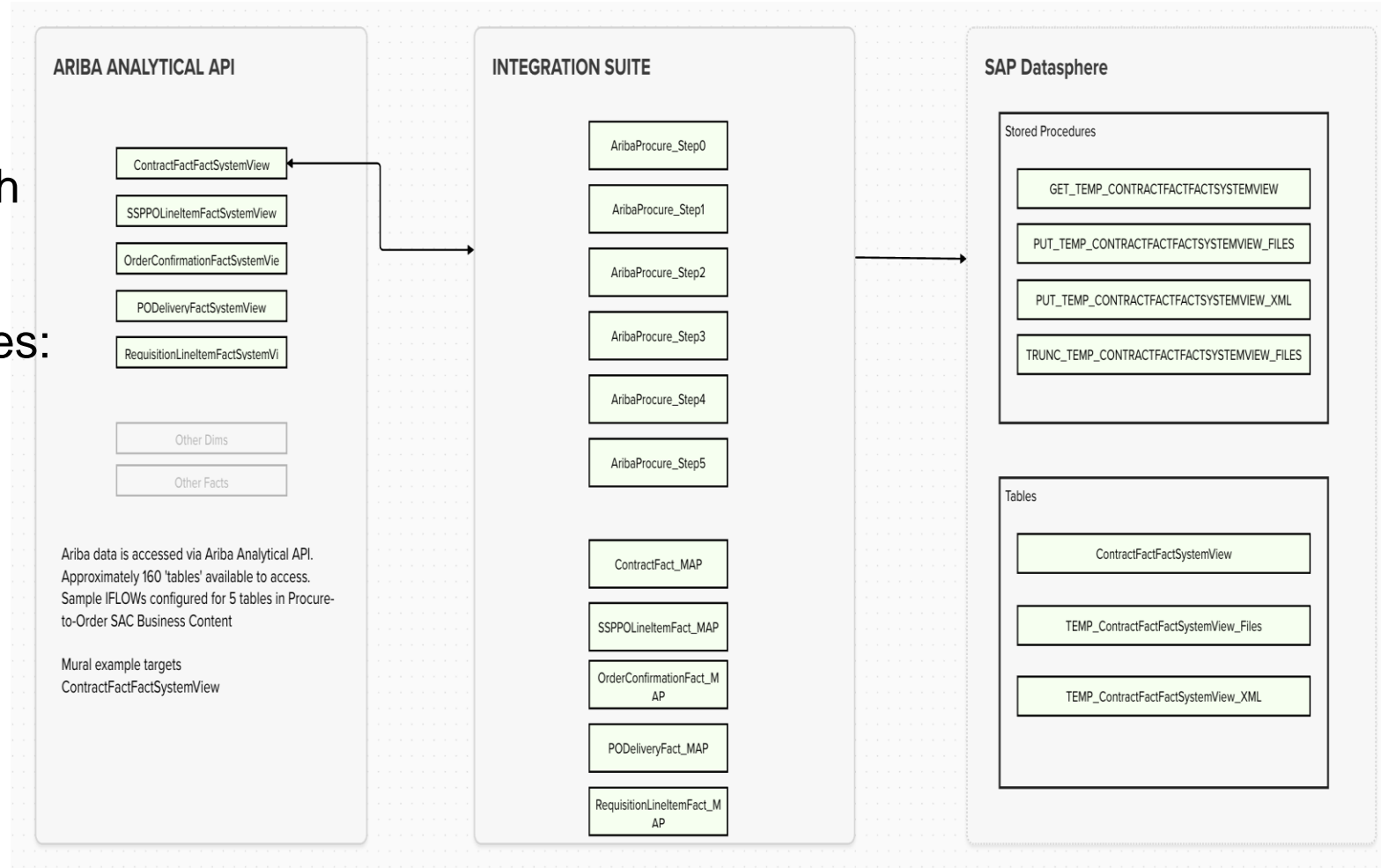
HANA Layer in Datasphere

Reviewing solution architecture –
Integration Suite communicates with
the following

Every targeted System View requires:

- Table
- Temp Table 1 (files)
- Temp Table 2 (xml)
- Procedure1 (put_temp_files)
- Procedure2 (get_temp_files)
- Procedure3 (trunc_temp_files)
- Procedure4 (put_temp_xml)

Scripts to create are available



Integration Flow Package Parameters

Flow: Step 1 -> Configure

Single screen to configure all points in Flow package:

- Ariba Token / Realm
- Target SystemView / To-From Dates
- Ensure Flows link to correct SQL Procs and Mapping files

Configure "AribaSpend_Step_1"

More

Type:	All Parameters
11_Token_Credential:	()
12_JDBC_Credential:	JDBC_Credetnial_Not_Supported_Yet
13_APIKEY:	()
14_Realm:	()
15_ViewTemplateName:	UNSPSCDimSystemView
16_FromDate:	2021-01-01T00:00:00Z
17_ToDate:	2021-01-31T23:55:31Z
18_MessageMap:	ref:Ariba_UNSPSCDimMap
19_SPROC_Files:	SAP_CONTENT#ARIBA_SPEND_ANALYSIS.PUT_TEMP_UNSPSCDIMSYSTEMVIEW_FILES
20_SPROC_Final:	SAP_CONTENT#ARIBA_SPEND_ANALYSIS.PUT_TEMP_UNSPSCDIMSYSTEMVIEW_XML
21_SPROC_GET_FILE:	SAP_CONTENT#ARIBA_SPEND_ANALYSIS.GET_TEMP_UNSPSCDIMSYSTEMVIEW_FILES
22_SPROC_TRN_FILE:	SAP_CONTENT#ARIBA_SPEND_ANALYSIS.TRUNC_TEMP_UNSPSCDIMSYSTEMVIEW_FILES
23_GrantType:	openapi_2lo
24_sleep_in_mins:	4
25_Split_Chunk_count:	50000
26_logger:	True
27_Token_URL:	https://api.ariba.com/v2/oauth/token
28_Jobs_URL:	https://openapi.ariba.com/api/analytics-reporting-job/v1/prod/jobs
29_jobStatus_URL:	https://openapi.ariba.com/api/analytics-reporting-jobresult/v1/prod/jobs

Ariba Token

Ariba API Key

Ariba Realm

Ariba System View (control API Job)

To And From Dates (control APIJob)

Integration Suite Mapping File name

Datasphere Stored Procedures:
19: PUT_TEMP_<tablename>_FILES
20: PUT_TEMP_<tablename>_XML
21: GET_TEMP_<tablename>_FILES
22: TRUNC_TEMP_<tablename>_FILES

Save Deploy Close

Integration Suite Demonstration

VIDEO DEMO Integration Flows Moving Data into Datasphere



**Datasphere:
Prepare HANA Objects**

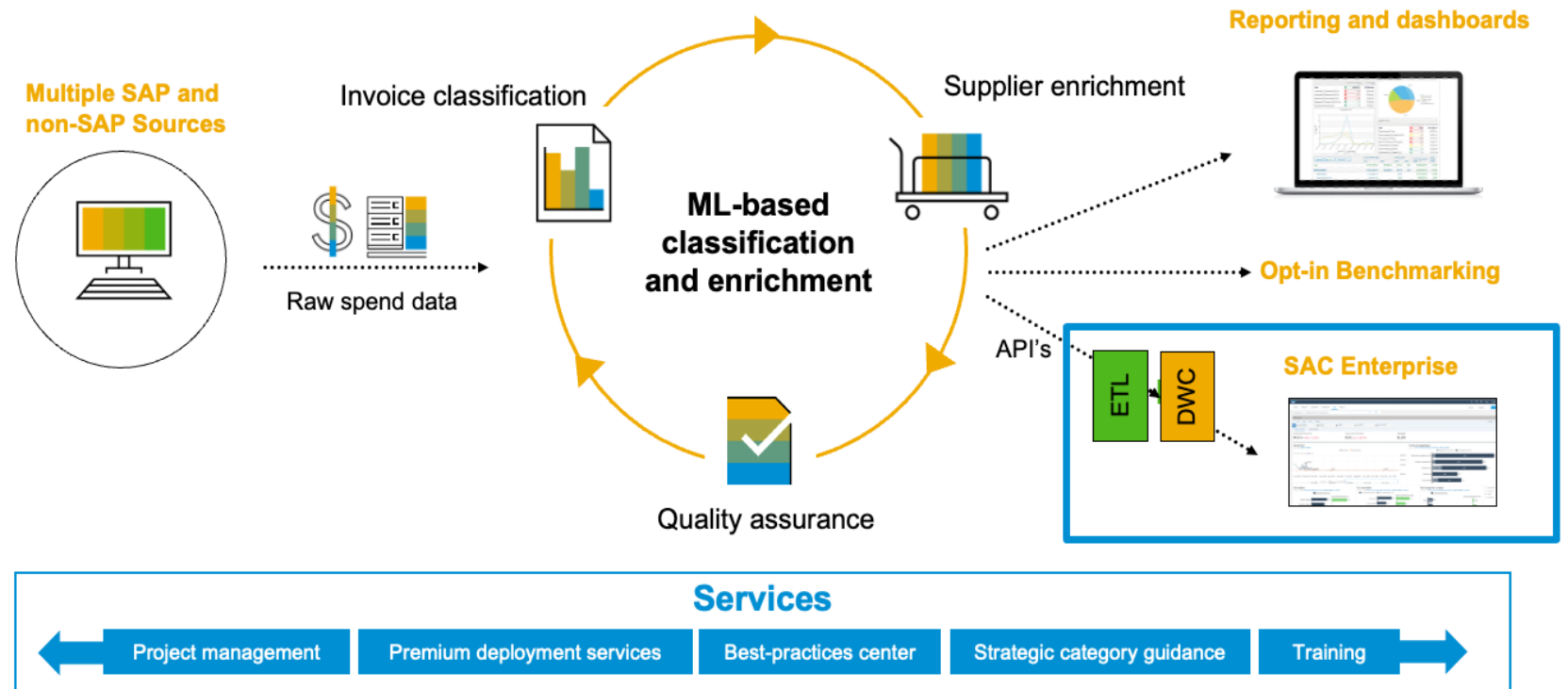
Available Integration Packages

Integration Package 1: DataspHERE Business Content – Spend Analysis

First Package using
Integration Flows - Ariba
Spend Analysis

- Spend is additional offering to Ariba
- Involves specific SAC stories on live connection to DataspHERE models
- Used as our Pilot phase to work through Integration Flow development

SAP Ariba Spend Analysis, advanced reporting and analytics add-on Content Starter kit for SAP Analytics Cloud and SAP Data Warehouse Cloud



© 2022 SAP SE or an SAP affiliate company. All rights reserved. | INTERNAL – SAP and Customers only

6

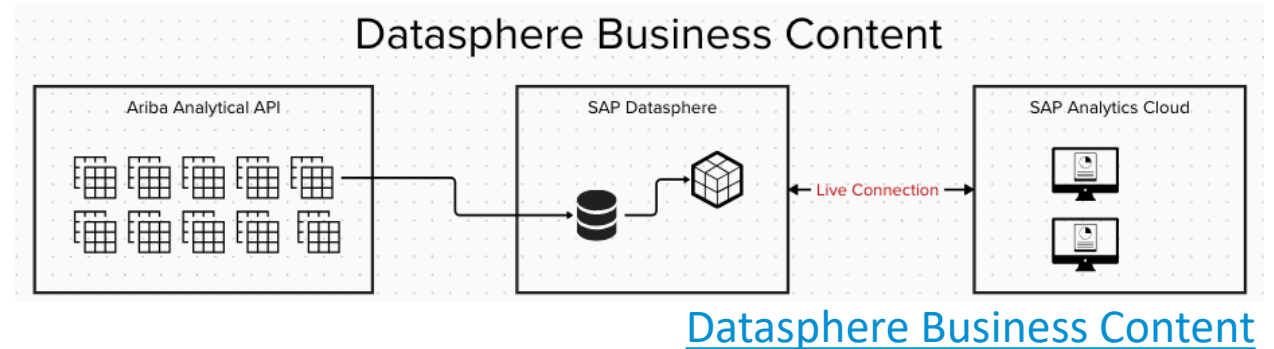
Datasphere Business Content

Downloadable Business Content:

- SAC Stories
- Datasphere Views and Model

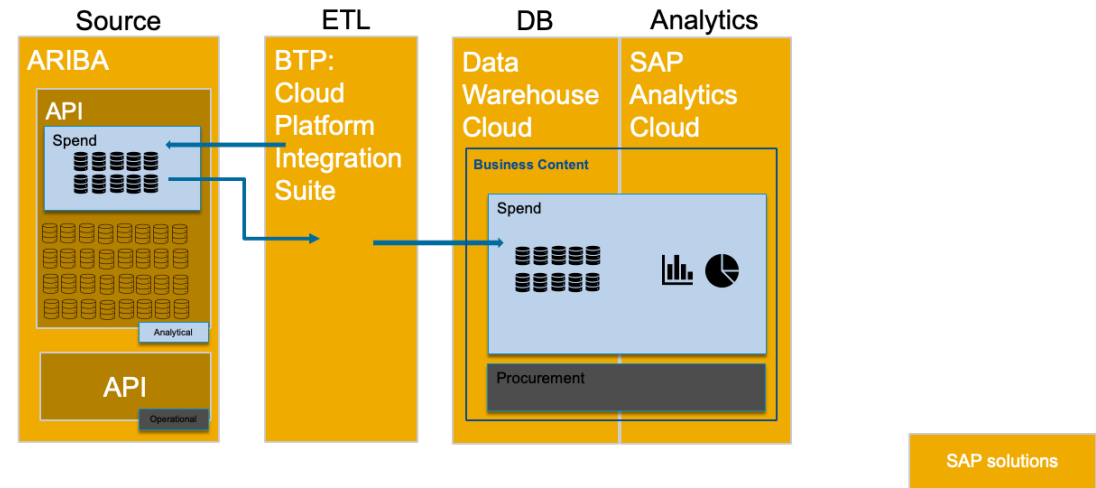
Need to build base tables and load with data to complete

Can use Integration Suite Flow package to accomplish



Spend Analytics – BTP Solution Overview

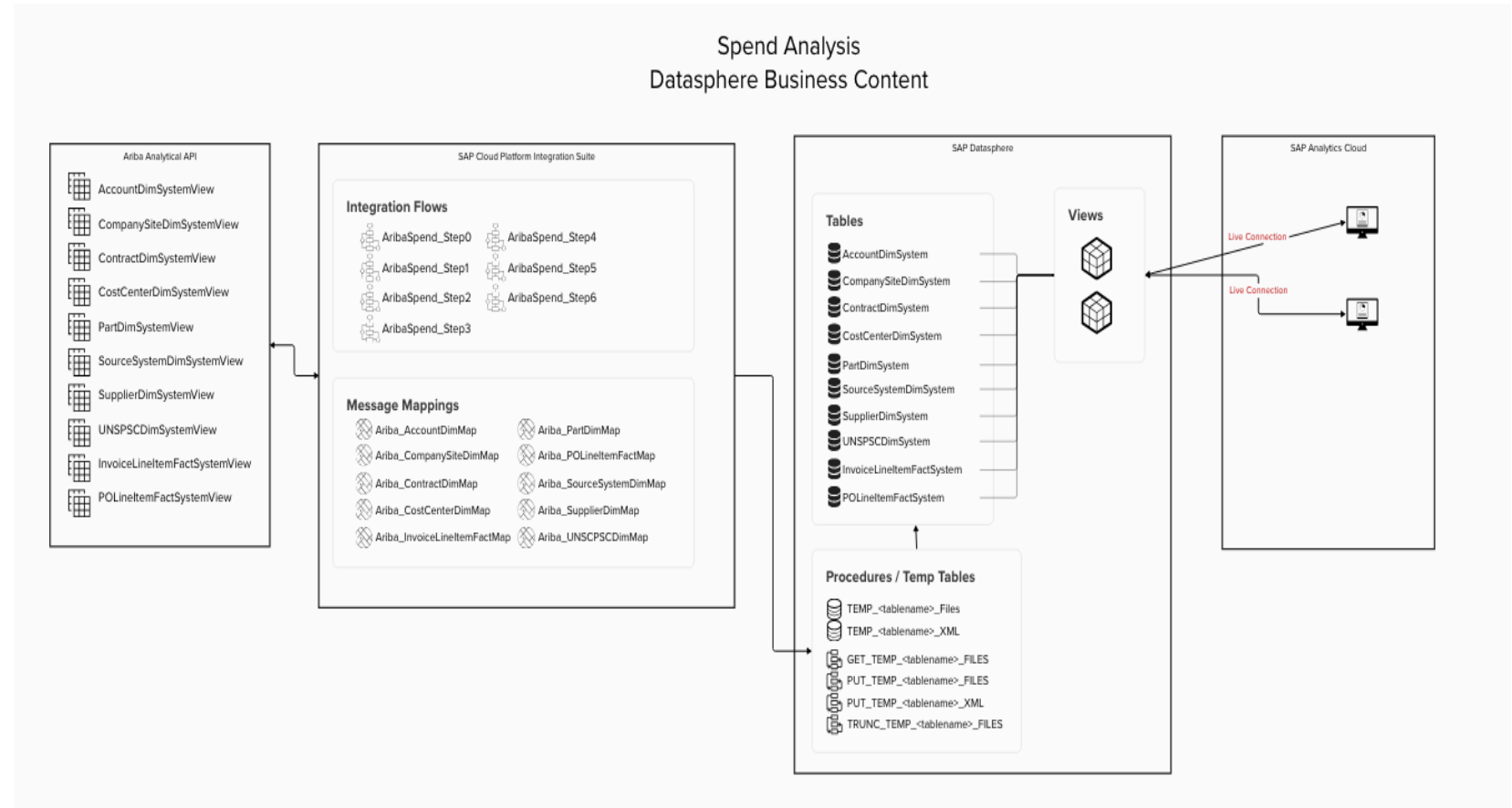
- Integration Flow targets 10 tables powering Spend Business Content
- Additional tables accessible via Ariba Analytical API – Possible to modify flow to connect



Datasphere Business Content / Integration Flows

Integration Suite Flow Package contents:

- 6 Flows (per earlier)
- 10 Mappings (1 per System View)
- HANA SQL Scripts
 - 1 script to create all base tables
 - 10 scripts (1 per system view) for Temp Tables and Stored Procedure creation



Integration Package 2: SAP Analytics Cloud Business Content Procure to Order

Downloadable Business Content:

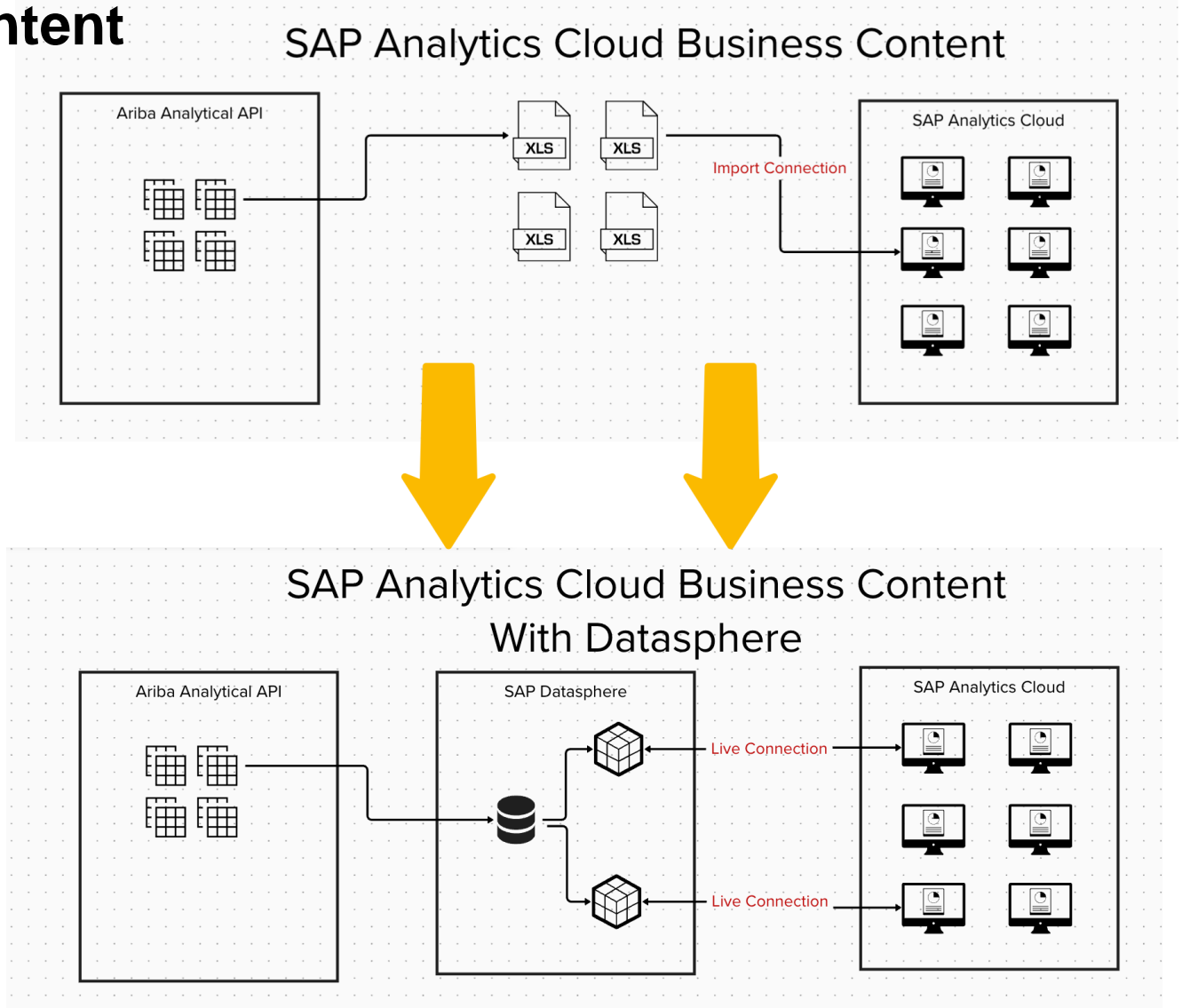
- SAC Stories

No Ariba connection in SAC – import data into system (csv/xls)

Reverse Engineer SAC models into Datasphere Tables/Views. Re-built SAC story off live connection

Used as second phase to expand reach and show we can target other Ariba System Views

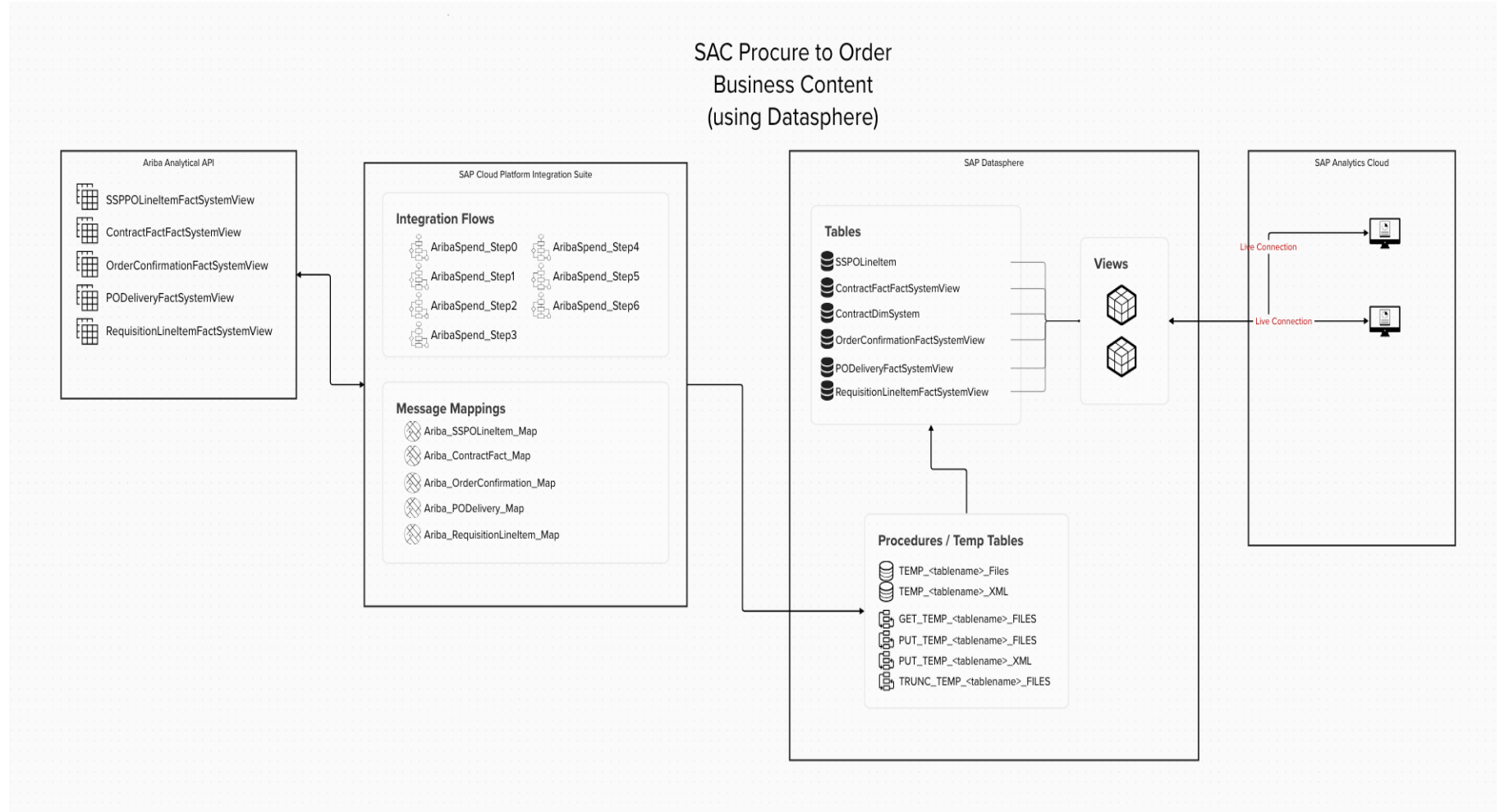
[SAC Business Content](#)



SAC Business Content / Integration Flows

Integration Suite Flow Package contents:

- 6 Flows (per earlier)
- 5 Mappings (1 per System View)
- HANA SQL Scripts
 - 1 script per base table (5 scripts)
 - 5 scripts (1 per system view) for Temp Tables and Stored Procedure creation



Customization Opportunity

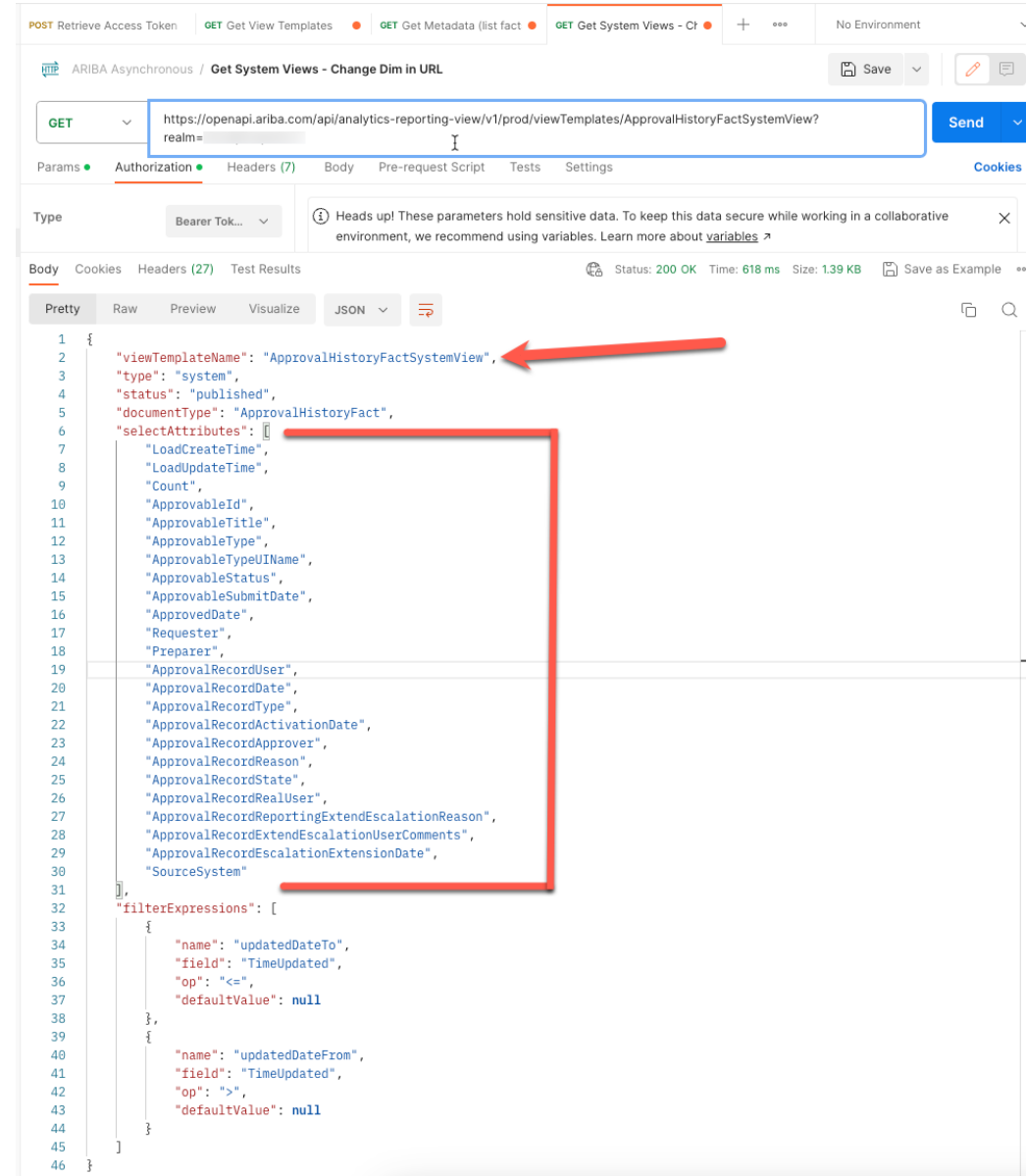
How to customize for any Ariba System View

6 Steps

1. Get Column Names from Ariba
2. HANA table creation script
3. HANA stored procedure creation script
4. Integration Suite Mapping: Request XSD
5. Integration Suite Mapping: Response XSD
6. Deploy / Map / Run Flow

Step 1 – Get Column Names

- Using Postman (or similar) – get the column names from target System View (slide 9)
- This Example will target “ApprovalHistoryFact”
- `https://openapi.riba.com/api/analytics-reporting-view/v1/prod/viewTemplates/ApprovalHistoryFactSystemView?realm=<YOURREALM>`
- Copy ‘selectAttributes’ from JSON response to use in scripts



Step 2 – HANA table creation script

- Use existing table create as template
- Replace table name and column names with target
 - CREATE COLUMN TABLE
" <schemaname> "." <tablename> " (
 - " <columnname> " nvarchar(5000),
- Example:
ApprovalHistoryFactSystemView

```
/*DROP TABLE SCRIPTS IF NEED TO RERUN TABLE CREATES*/
/*
DROP TABLE "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."PODeliveryFactSystemView";
*/

CREATE COLUMN TABLE "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."PODeliveryFactSystemView"(
  "LoadCreateTime" NVARCHAR(5000),
  "LoadUpdateTime" NVARCHAR(5000),
  "POId" NVARCHAR(5000),
  "OrderID" NVARCHAR(5000),
  "POName" NVARCHAR(5000),
  "ProcurementUnit" NVARCHAR(5000),
  "Amount" NVARCHAR(5000),
  "ConfirmationTime" NVARCHAR(5000),
  "DeliveryTime" NVARCHAR(5000),
  "ReceiptTime" NVARCHAR(5000),
  "InvoiceTime" NVARCHAR(5000),
  "OnTimeDeliveryShip" NVARCHAR(5000),
  "OnTimeDeliveryReceipt" NVARCHAR(5000),
  "OnTimeOrLate" NVARCHAR(5000),
  "BackOrderedItems" NVARCHAR(5000),
  "RejectedItems" NVARCHAR(5000),
  "SubstitutedItems" NVARCHAR(5000),
  "OrderConfirmation" NVARCHAR(5000),
  "AdvancedShipNotice" NVARCHAR(5000),
  "Receipt" NVARCHAR(5000),
  "OrderedDate" NVARCHAR(5000),
  "NeedByDate" NVARCHAR(5000),
  "Supplier" NVARCHAR(5000),
  "Requester" NVARCHAR(5000),
  "SourceSystem" NVARCHAR(5000)
);

--ALTER TABLE "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."PODeliveryFactSystemView"
--ADD PRIMARY KEY("","","");
```

ResponseFieldNames

```
"LoadCreateTime",
"LoadUpdateTime",
"Count",
"ApprovableId",
"ApprovableTitle",
"ApprovableType",
"ApprovableTypeUIName",
"ApprovableStatus",
"ApprovableSubmitDate",
"ApprovedDate",
"Requester",
"Preparer",
"ApprovalRecordUser",
"ApprovalRecordDate",
"ApprovalRecordType",
"ApprovalRecordActivationDate",
"ApprovalRecordApprover",
"ApprovalRecordReason",
"ApprovalRecordState",
"ApprovalRecordRealUser",
"ApprovalRecordReportingExtendEscalationReason",
"ApprovalRecordExtendEscalationUserComments",
"ApprovalRecordEscalationExtensionDate",
"SourceSystem"
```

Step 3 – HANA stored procedure creation script

- Use existing Stored Procedure creation script as template
- Replace proc names **<name>** with new
 - COLUMN TABLE: TEMP_<name>_FILES
 - COLUMN TABLE: TEMP_<name>_XML
 - PUT_TEMP_<name>_FILES
 - GET_TEMP_<name>_FILES
 - TRUNC_TEMP_<name>_FILES
 - PUT_TEMP_<name>_XML
- Replace Columns in PUT_TEMP_<name>_XML script
 - “column” NVARCHAR(5000) PATH ‘column’,
 - NOTE double & single quotes
- Example: ApprovalHistoryFactSystemView

```
CREATE COLUMN TABLE "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES" (JOB_KEY varchar(20), JOB_FILE_NAME CLOB);
CREATE COLUMN TABLE "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_XML" (FILENAME varchar(20), FILECONTENT CLOB);

CREATE OR REPLACE PROCEDURE SAP_CONTENT#ARIBA_SPEND_ANALYSIS.PUT_TEMP_PODeliveryFactSystemView_FILES (
  IN JOB_KEY varchar(20),
  IN JOB_FILE_NAME CLOB)
LANGUAGE SQLSCRIPT AS
BEGIN
  DECLARE counter int;
  DECLARE V_JFN_SOURCE CLOB;
  DECLARE V_JFN_TARGET CLOB;

  SELECT count(*) INTO counter FROM "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES";
  IF :counter = 0
  THEN
    INSERT INTO "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES" VALUES ( JOB_KEY, JOB_FILE_NAME);
  ELSE
    SELECT top 1 JOB_FILE_NAME INTO V_JFN_SOURCE FROM "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES";
    V_JFN_TARGET = CONCAT(V_JFN_SOURCE, JOB_FILE_NAME);
    Truncate table "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES";
    INSERT INTO "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES" VALUES ( JOB_KEY, V_JFN_TARGET);
  END IF;
END;

CREATE OR REPLACE PROCEDURE SAP_CONTENT#ARIBA_SPEND_ANALYSIS.GET_TEMP_PODeliveryFactSystemView_FILES ()
LANGUAGE SQLSCRIPT AS
BEGIN
  SELECT "JOB_KEY","JOB_FILE_NAME" FROM "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES";
END;

CREATE OR REPLACE PROCEDURE SAP_CONTENT#ARIBA_SPEND_ANALYSIS.TRUNC_TEMP_PODeliveryFactSystemView_FILES ()
LANGUAGE SQLSCRIPT AS
BEGIN
  Truncate Table "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_FILES";
END;

CREATE OR REPLACE PROCEDURE SAP_CONTENT#ARIBA_SPEND_ANALYSIS.PUT_TEMP_PODeliveryFactSystemView_XML (
  IN FILENAME varchar(20),
  IN FILECONTENT CLOB)
LANGUAGE SQLSCRIPT AS
BEGIN
  INSERT INTO "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_XML" VALUES (
    FILENAME,
    FILECONTENT);

  INSERT INTO "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."PODeliveryFactSystemView" (
    SELECT * FROM
      XMLTABLE('resultset/row' PASSING
        SAP_CONTENT#ARIBA_SPEND_ANALYSIS."TEMP_PODeliveryFactSystemView_XML"."FILECONTENT"
        COLUMNS
          "LoadCreateTime" NVARCHAR(5000) PATH 'LoadCreateTime',
          "LoadUpdateTime" NVARCHAR(5000) PATH 'LoadUpdateTime',
          "POId" NVARCHAR(5000) PATH 'POId',
          "OrderID" NVARCHAR(5000) PATH 'OrderID',
          "POName" NVARCHAR(5000) PATH 'POName',
          "ProcurementUnit" NVARCHAR(5000) PATH 'ProcurementUnit',
          "Amount" NVARCHAR(5000) PATH 'Amount',
          "ConfirmationTime" NVARCHAR(5000) PATH 'ConfirmationTime',
          "DeliveryTime" NVARCHAR(5000) PATH 'DeliveryTime',
          "ReceiptTime" NVARCHAR(5000) PATH 'ReceiptTime',
          "InvoiceTime" NVARCHAR(5000) PATH 'InvoiceTime',
          "OnTimeDeliveryShip" NVARCHAR(5000) PATH 'OnTimeDeliveryShip',
          "OnTimeDeliveryReceipt" NVARCHAR(5000) PATH 'OnTimeDeliveryReceipt',
          "OnTimeOrLate" NVARCHAR(5000) PATH 'OnTimeOrLate',
          "BackOrderedItems" NVARCHAR(5000) PATH 'BackOrderedItems',
          "RejectedItems" NVARCHAR(5000) PATH 'RejectedItems',
          "SubstitutedItems" NVARCHAR(5000) PATH 'SubstitutedItems',
          "OrderConfirmation" NVARCHAR(5000) PATH 'OrderConfirmation',
          "AdvancedShipNotice" NVARCHAR(5000) PATH 'AdvancedShipNotice',
          "Receipt" NVARCHAR(5000) PATH 'Receipt',
          "OrderedDate" NVARCHAR(5000) PATH 'OrderedDate',
          "NeedByDate" NVARCHAR(5000) PATH 'NeedByDate',
          "Supplier" NVARCHAR(5000) PATH 'Supplier',
          "Requester" NVARCHAR(5000) PATH 'Requester',
          "SourceSystem" NVARCHAR(5000) PATH 'SourceSystem'
        ) as XTABLE
  );
  Truncate Table "SAP_CONTENT#ARIBA_SPEND_ANALYSIS"."TEMP_PODeliveryFactSystemView_XML";
END;
```

ResponseFieldNames

- "LoadCreateTime",
- "LoadUpdateTime",
- "Count",
- "ApprovableId",
- "ApprovableTitle",
- "ApprovableType",
- "ApprovableTypeUName",
- "ApprovableStatus",
- "ApprovableSubmitDate",
- "ApprovedDate",
- "Requester",
- "Preparer",
- "ApprovalRecordUser",
- "ApprovalRecordDate",
- "ApprovalRecordType",
- "ApprovalRecordActivationDate",
- "ApprovalRecordApprover",
- "ApprovalRecordReason",
- "ApprovalRecordState",
- "ApprovalRecordRealUser",
- "ApprovalRecordReportingExtendEscalationReason",
- "ApprovalRecordExtendEscalationUserComments",
- "ApprovalRecordEscalationExtensionDate",
- "SourceSystem"

Step 4 – Integration Suite Mapping: Request XSD

- Use existing Request XSD as template
- Replace element names **<name>** with new
- Pattern:

`<xs:element type="xs:string" name="<name>"/>`

The screenshot displays an XML editor window titled 'ApprovalHistoryFactSystemView_request.xsd'. The XML content is as follows:

```
1 <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <xs:element name="Root">
3     <xs:complexType>
4       <xs:sequence>
5         <xs:element name="Element" maxOccurs="unbounded" minOccurs="0">
6           <xs:complexType>
7             <xs:sequence>
8               <xs:element type="xs:string" name="LoadCreateTime"/>
9               <xs:element type="xs:string" name="LoadUpdateTime"/>
10              <xs:element type="xs:string" name="POId"/>
11              <xs:element type="xs:string" name="OrderID"/>
12              <xs:element type="xs:string" name="POName"/>
13              <xs:element type="xs:string" name="ProcurementUnit"/>
14              <xs:element type="xs:string" name="Amount"/>
15              <xs:element type="xs:string" name="ConfirmationTime"/>
16              <xs:element type="xs:string" name="DeliveryTime"/>
17              <xs:element type="xs:string" name="ReceiptTime"/>
18              <xs:element type="xs:string" name="InvoiceTime"/>
19              <xs:element type="xs:string" name="OnTimeDeliveryShip"/>
20              <xs:element type="xs:string" name="OnTimeDeliveryReceipt"/>
21              <xs:element type="xs:string" name="OnTimeOrLate"/>
22              <xs:element type="xs:string" name="BackOrderedItems"/>
23              <xs:element type="xs:string" name="RejectedItems"/>
24              <xs:element type="xs:string" name="SubstitutedItems"/>
25              <xs:element type="xs:string" name="OrderConfirmation"/>
26              <xs:element type="xs:string" name="AdvancedShipNotice"/>
27              <xs:element type="xs:string" name="Receipt"/>
28              <xs:element type="xs:string" name="OrderedDate"/>
29              <xs:element type="xs:string" name="NeedByDate"/>
30              <xs:element type="xs:string" name="Supplier"/>
31              <xs:element type="xs:string" name="Requester"/>
32              <xs:element type="xs:string" name="SourceSystem"/>
33            </xs:sequence>
34          </xs:complexType>
35        </xs:element>
36      </xs:sequence>
37    </xs:complexType>
38  </xs:element>
39 </xs:schema>
```

On the right, a smaller window titled 'ResponseFieldNames' shows a list of field names to be mapped to the elements in the XSD. The list is as follows:

```
1 "LoadCreateTime",
2 "LoadUpdateTime",
3 "Count",
4 "Approvable",
5 "ApprovableTitle",
6 "ApprovableType",
7 "ApprovableTypeUIName",
8 "ApprovableStatus",
9 "ApprovableSubmitDate",
10 "ApprovedDate",
11 "Requester",
12 "Preparer",
13 "ApprovalRecordUser",
14 "ApprovalRecordDate",
15 "ApprovalRecordType",
16 "ApprovalRecordActivationDate",
17 "ApprovalRecordApprover",
18 "ApprovalRecordReason",
19 "ApprovalRecordState",
20 "ApprovalRecordRealUser",
21 "ApprovalRecordReportingExtendEscalationReason",
22 "ApprovalRecordExtendEscalationUserComments",
23 "ApprovalRecordEscalationExtensionDate",
24 "SourceSystem"
```

A red arrow points from the 'name' attribute in the XSD to the list of field names.

Step 5 – Integration Suite Mapping: Response XSD

- Use existing Response XSD as template
- Replace element names **<name>** with new
- Pattern:

`<xs:element name="<name>" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />`

```
1 <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <xs:element name="resultset">
3     <xs:complexType>
4       <xs:sequence>
5         <xs:element name="row" maxOccurs="unbounded" minOccurs="0">
6           <xs:complexType>
7             <xs:sequence>
8               <xs:element name="LoadCreateTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
9               <xs:element name="LoadUpdateTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
10              <xs:element name="POId" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
11              <xs:element name="OrderID" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
12              <xs:element name="POName" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
13              <xs:element name="ProcurementUnit" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
14              <xs:element name="Amount" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
15              <xs:element name="ConfirmationTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
16              <xs:element name="DeliveryTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
17              <xs:element name="ReceiptTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
18              <xs:element name="InvoiceTime" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
19              <xs:element name="OnTimeDeliveryShip" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
20              <xs:element name="OnTimeDeliveryReceipt" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
21              <xs:element name="OnTimeOrLate" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
22              <xs:element name="BackOrderedItems" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
23              <xs:element name="RejectedItems" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
24              <xs:element name="SubstitutedItems" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
25              <xs:element name="OrderConfirmation" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
26              <xs:element name="AdvancedShipNotice" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
27              <xs:element name="Receipt" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
28              <xs:element name="OrderedDate" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
29              <xs:element name="NeedByDate" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
30              <xs:element name="Supplier" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
31              <xs:element name="Requester" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
32              <xs:element name="SourceSystem" minOccurs="0" maxOccurs="1" nillable="true" type="xs:string" />
33            </xs:sequence>
34          </xs:complexType>
35        </xs:element>
36      </xs:sequence>
37    </xs:complexType>
38  </xs:element>
39 </xs:schema>
```

ResponseFieldNames
Edited

- "LoadCreateTime",
- "LoadUpdateTime",
- "Count",
- "Approvable",
- "ApprovableTitle",
- "ApprovableType",
- "ApprovableTypeUIName",
- "ApprovableStatus",
- "ApprovableSubmitDate",
- "ApprovedDate",
- "Requester",
- "Preparer",
- "ApprovalRecordUser",
- "ApprovalRecordDate",
- "ApprovalRecordType",
- "ApprovalRecordActivationDate",
- "ApprovalRecordApprover",
- "ApprovalRecordReason",
- "ApprovalRecordState",
- "ApprovalRecordRealUser",
- "ApprovalRecordReportingExtendEscalationReason",
- "ApprovalRecordExtendEscalationUserComments",
- "ApprovalRecordEscalationExtensionDate",
- "SourceSystem"

Lines: 24 Characters: 554 Location: 554 Line: 24

Lines: 39 Characters: 2,943 Location: 2,730 Line: 32

How to customize for any Ariba System View

1. Run HANA Scripts
 - a. Table Creation Scripts (temp and target)
 - b. Stored Procedure Scripts (4)
2. Create new Integration Suite Mapping File
 - a. Load source (Request) and target (Response) XSD files
 - b. Map all Request to Response columns in GUI
3. Run Integration Suite Flows
 - a. Deploy new Mapping (and flows if in new package)
 - b. Change Parameters in Step 1
 - c. Deploy Flows and kick off Step 0

Helpful Links

- SAP Discovery Mission: [link](#)
- Integration Flows / SQL Scripts (parameterized version): [link](#)
- SAP Datasphere Business Content: [link](#)
- SAP Analytic Cloud Business Content: [link](#)
- Discovery Center – SAP Cloud Platform Integration Suite: [link](#)
- Discovery Center – SAP Datasphere: [link](#)
- Discovery Center – SAP Analytics Cloud: [link](#)

Thank you.

Contact information:

Todd Hanna
todd.hanna@sap.com

