ASCEND.IO

Custom vs. Prebuilt API Connectors

Benefits, Limitations & What Makes the "Perfect Connector"

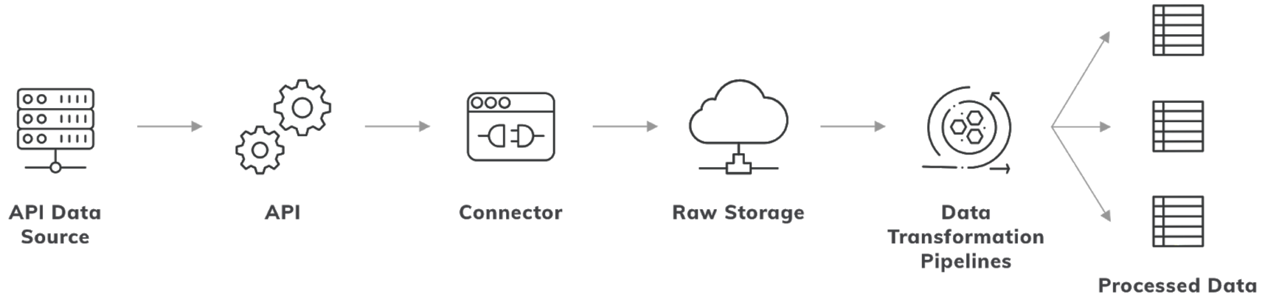
Introduction

APIs are a unique category of data source, in that they are each a snowflake with their own set of connection criteria, imposed connection limitations, and schemas. This makes them expensive to build and maintain, with little leverage across APIs. Some progress has been made with vendors who provide "out of the box" Connectors, but given how expensive they are to maintain, you will find "out of the box" Connectors are primarily focused on a small number of highly-used APIs (i.e. Facebook, Google Analytics). Anything outside of these highly-used connectors requires a different strategy, and is likely why you are here.

In this paper, we aim to help you understand the following:

1. The anatomy of an API Connector
2. The challenges you should anticipate
3. The pro's and con's of the various approaches
4. Criteria for evaluating platforms with API Connectors.

The Anatomy of an API Workflow



API SOURCE DATA

External data store (or endpoint) managed by a 3rd party that can be free/public or paid. API Source Data has three key characteristics: 1) It is not designed to be queried directly as a system of record, 2) limits are frequently put on how often you can query and how much data can be returned and 3) the data from the query is returned in a semi-structure JSON format that requires further transformations to be usable.

API

Access interface made available by the owner of the external data store. API documentation is typically provided around the interface itself as well as the data that is returned from the API, however the quality of documentation can vary wildly.

CONNECTOR

Software code that connects to the API to make requests for data. Connectors generally come in two flavors: 1) Prebuilt from a vendor where the software code itself is fully managed and kept up to date by the vendor, or 2) Custom, which uses a 3rd Party platform to minimize the amount of code needed to be written and maintained.

RAW STORAGE
Typically a blog store to house the raw data as it is returned from the API, however recent advances on the Data Warehouse side are making it viable to store this raw data directly into the Data Warehouse. For cost and ease of data transformation, the common pattern is still the blob store.

DATA TRANSFORMATION PIPELINES
Cascading transforms that will clean and standardize the schema, join the data with other sources to provide richer semantics, and create pre-aggregated views for high volume data. In addition, you will need to create a "system of record" that includes all historical data needed for reporting.

PROCESSED DATA
Finally, we have nice relational data from the Pipelines for users to query! This typically lands inside a Data Warehouse, or for Data Science/ML applications, accessed in a notebook directly from a blob store.

Why are APIs hard?

Given our years of experience with APIs, we have found the following to be the primary challenges in dealing with this type of data:

AUTHENTICATION DIFFERENCES	● Standard and non-standard authentication schemes ● Even standard schemes vary in complexity (basic auth vs oauth)
RATE LIMITS	● Many APIs do not have rate limits that are amenable to pulling data out for export use cases (especially for backills) ● There could be a separate export API where the API structure is different (job-based, async API) and the return schema of the export API is often different than the normal endpoints
API DATA CAN BE REALLY RAW/UNUSABLE	● It's a single stream of data that contains data for multiple tables (semi-structured) ● Columns are poorly documented ● Every API schema is a snowflake/normalization ● It needs enriching from other API endpoints
API "BREAKS"	● Frequent version changes or deprecation ● API Parameters Change ● Unannounced Schema Evolution ● Bad Data Injection

HISTORICAL INFORMATION IS OFTEN LOST	● Most API providers do not provide historical information, only the current state, requiring you to keep your own archives for higher fidelity analysis ● Current state data often has less analytical value than data changing over time

Different Approaches to APIs

Prebuilt Connectors

All Prebuilt Connectors look to automate the “Connector” part of the diagram as much as possible. Users will typically only see a basic UI that asks for information required to make the API connection itself (like credentials). In the background, the Connector will handle the following core capabilities to various degrees of success:

- Connection parameters
- Credentials
- Retry/error handling
- API specific parameters (rate limits, payload size limits, etc)
- Schema specific endpoints
- Incremental logic
- Schema evolution

These types of connectors will typically drop the raw API payload into a blob store or warehouse, at which point, work needs to be done to convert the payload into usable, analytic format. This work can be pretty significant, thus the rise of Data-as-a-Service vendors.

In considering Prebuilt Connectors, we suggest evaluating two key dimensions:

- **What percent of your APIs are in a fully supported library from the vendor?**

  Many vendors will list “community supported” APIs, which means you are responsible for any issues. If a high percentage of your APIs are found in a fully supported library from a vendor, the vendor would be a strong candidate to invest in further evaluation. If you have a fair number of un-supported APIs, we suggest evaluating platforms that support Custom Connectors.

- **How much transformation work is needed to make the data analytic ready?**

  Be sure to evaluate the gap between the raw data coming from the Connector and data requirements for analytic purposes. If there is a significant delta you should a) make sure the Prebuilt Connector vendor enables you to develop/maintain these transformations or b) have another technology product available for this purpose. If neither are available options in a Prebuilt Connector, you may want to consider Data-as-a-Service vendors.

Data-as-a-Service

Data-as-a-Service vendors provide additional processing of the data from a small subset of APIs to make it more usable for the data consumer. This processing can include services such as:

- Json Parsing
- Data Cleansing
- Transform Decisions

Sometimes the data schema is exactly what you need, but most of the time it's not. Frequent gaps include missing fields, lack of denormalization leading to tedious joins or data drifts (current state not being properly snapshotted), or needing to make other API calls to further enrich the data.

In considering Data-as-a-Service vendors, we suggest you focus considerable effort to ensure that the data output is *exactly* what you need. Otherwise you will find yourself in a similar situation as any Prebuilt Connector where you will need to write your own additional transformations to make the data analytic ready.

Custom Connectors

The world of data APIs is enormous, and only a handful can be found as a Prebuilt Connector or provided through a Data-as-a-Service vendor. For the vast majority of these APIs, coding will be required to properly interface with the API. For software engineers, building a fully custom API connector is a possible alternative; however, the complexity of the code, and the ongoing maintenance required to handle an ever changing API, is too big of a burden for most. Instead, **look for a robust platform that provides the ease and automation of Prebuilt Connectors but the flexibility to transform the data at will. This will greatly reduce the amount of code and limit maintenance burdens.**

Some of the key areas that a robust custom connector platform will provide include:

- Scheduling/managing API connections/calls for incremental data.
- Spinning up/down all necessary infrastructure to accommodate large or frequent API calls.
- Error handling / retry management.
- Auto-running downstream transforms.
- Converting data / loading to blob store / warehouse.

If done well, the developer is left with just a small amount of manageable code specific to the individual API specifications, typically related to the "fetch" portion of the API.

The “Perfect Connector” Platform

The reality most people face when attempting to connect and ingest data from APIs is the following:

- Prebuilt Connectors cover a very small percentage of the desired APIs
- Data-as-a-Service vendors cover an even smaller percentage, and frequently do not provide all the required data
- ***Custom Connectors enable connections to any API, and thus become a core part of any data architecture/strategy***.

A perfect connector platform will have the following capabilities :

1. **A core architecture built to abstract away all code from the user to run/manage/scale connecting to and ingesting from the API, leaving the developer with a small amount of manageable code specific to the individual API specifications.**
2. **A set of Prebuilt Connectors for popular APIs that remove all coding from the user but gives access to the raw API payload.**
3. **A fully integrated/GUI based transform capability that allows the user to easily transform the raw API payload into analytic-ready data.**

About Ascend

Ascend provides the world’s first Autonomous Dataflow Service, enabling data engineering to build, scale, and operate continuously optimized, Apache Spark-based pipelines with 85% less code. Running natively in Microsoft Azure, Amazon Web Services, and Google Cloud Platform, Ascend combines declarative configurations and automation to manage the underlying cloud infrastructure, optimize pipelines, and eliminate maintenance across the entire data lifecycle. For more information about Ascend, visit www.ascend.io.