JESTA I.S.

Enterprise Software for Retailers,
e-Tailers and Wholesalers

WHITE PAPER:

REST API for Fast and Intuitive Integrations

Written by Robert Baron, Systems Architect

info@jestais.com
www.jestais.com

TABLE OF CONTENTS

PG. 3 ABSTRACT

PG. 3 INTRODUCTION

PG. 5 SOLUTION

PG. 8 SUCCESS STORIES

PG. 9 CONCLUSION

PG. 10 CONTACT US

ABSTRACT

This white paper describes Jesta's REST-based API. It elaborates on the reasons why the API was built, the architectural design decisions made during development, the API's features and capabilities, and the problems that the API is solving. The REST API has been used in several integrations for many Jesta customer implementations. Success stories are included on page 8.

INTRODUCTION

Jesta has a robust suite of ERP products that drives omnichannel journeys from product design to direct-to-consumer deliveries. Our customers are global retailers, wholesalers and brand manufacturers of all sizes and verticals. When a new client implements one or more of our products, data integration is often required between the client's ERP and Jesta's products. Some of the data needs to be synced in real time.

Real-Time Data Syncing

- E-commerce sales to Jesta's Omnichannel
- E-commerce inventory to Jesta's POS and Merchandising
- Warehouse shipment updates to Jesta's Omnichannel

Hourly Data Syncing

- Warehouse inventory to Jesta's Merchandising

Daily Data Syncing

- POS transactions to client's CRM
- Human Resource Information System (HRIS) employees list to Jesta's POS

Jesta needed a standard solution for web services and set out to build a RESTful API to expose the full functionality of our products

Many years ago, when most ERP systems were running on premise at a client's head office via the same LAN, integration could be easily achieved through inbound and outbound database tables. The exchange of real-time data was attained by running inbound and outbound processes every few minutes. We could bridge the gap between various database technologies using a standard like ODBC or by using flat files.

Over the years, many retailers have shifted from an on-premise ERP to one in the cloud (whether public or managed). The delocalization of ERPs made the use of inbound and outbound database tables less suitable because they required the set up of a VPN to access the database over the WAN. Often Internet speed and latency made the exchange of data painful.

Clients have, however, been asking for even more real-time data. Running inbound and outbound database processes every few minutes isn't suitable for many business processes any more. For example, when a customer submits an order on a client's e-commerce site, the e-commerce application must be able to check inventory availability and immediately create a sales order in Jesta's system. When the customer's order is accepted, stock must be adjusted accordingly in Jesta's Merchandising or POS system.

To respond to these client demands, Jesta started to build various web services. Several sets of web services have been built over the years based on ad-hoc specifications using different technologies (WSDL, WCF, TCP, ASP.NET Web API, etc. in Java, .NET, JavaScript, etc.); they were hosted on various web servers (IIS, Tomcat, WebLogic, etc.), including some homemade servers.

Eventually it became clear that Jesta needed a standard solution for these web services and we set out to build a RESTful API to expose the full functionality of all of our products. The primary purpose of our API is to make it easier and faster to integrate with our clients' ERPs, while also supporting more effective integration between our own applications.

SOLUTION

Architecture

Our API architecture is based on two major design decisions:

- OData Standard
- Rapid Development Environment

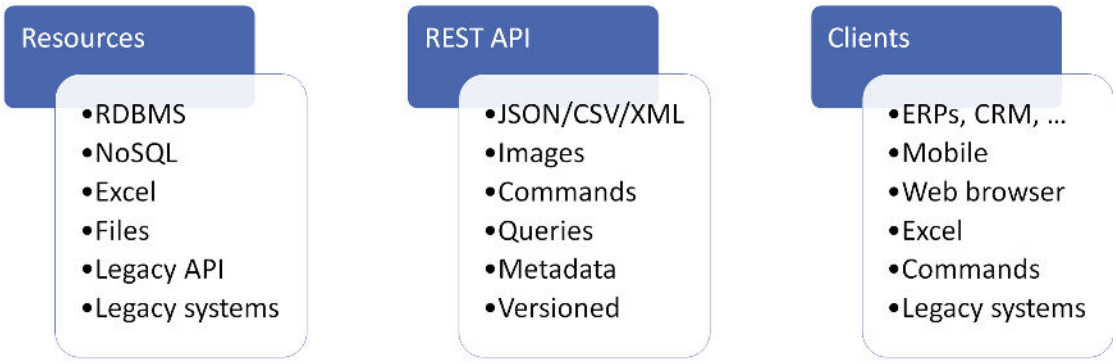
First, we wanted a common, consistent syntax and semantic for our REST API. The OData standard consists of just that. It provides a well-defined syntax and semantic, specifies how to expose metadata, is technology agnostic, and supports all data formats; SOAP-based web services support XML exclusively. As we implement new API integrations for our clients, we'll continue to expand our base set of OData features.

A second challenge is that every client's integration is custom and often requires specific code to be written. Customization often means creating a new database view or modifying an existing one, adding columns to a database table or creating a new stored database procedure. All of it then has to be exposed through our API for integration.

While common code can be reused from one integration to another, it inevitably needs to be modified, tested and debugged. Writing code to expose all tables wouldn't suffice because customization may very well require the addition and/or modification of database objects.

We could've designed our API as many other companies do and expose all of our system data ready for integration. In addition to being a major challenge given the tens of thousands of database objects that we have, this would have provided very little help for integration. The reality is that every integration is custom. Every client has unique needs and this requires that APIs be built specific to each of them.

Key Capabilities



Jesta's REST API exposes data resources and functionality, and allows clients to implement these resources with read, create, update and delete (CRUD) operations.

Database objects from all major Relational Database Management Systems (RDBMS) are automatically exposed using our DAL/API endpoint code generation tool without having to write any code. We only expose the database objects needed by each client.

Our API is primarily for integration and is, therefore, customized for each client. With custom code, we can expose all data resources virtually ranging from NoSQL databases to Excel files and flat files. We can also call into a client's legacy system using their API or SDK exposed via API actions endpoint URLs.

Working with JSON

Data is for the most part exposed in the JSON format, but other formats are available. For example, while one mobile client retrieving a product image may prefer to receive a binary image, another client may prefer the image encoded in base 64 format.

The OData standard has its own query language that can be used by clients to filter a collection of records based on field values. The filters can be used to select specific fields, page through records, sort them or automatically expand records' child (detailed) records. For efficiency, this query language takes advantage of the query capability of the underlying data resource (SQL in the case of a RDBMS data resource).

Discoverability

Metadata is available by sending a GET request at the end of a specific API URL. The metadata consists of an XML document that describes every endpoint of the API, how to call every custom action

and function, all the available record sets and their fields each with its name and data type. Because metadata is surfaced in the API, clients can discover the API capability and act accordingly, and also check for the availability of functionalities and/or data resources.

API consumers can be any HTTP-capable application. Many ERPs provide extensibility points where custom actions and functions can be implemented in various programming languages. Today, all of these programming languages can send HTTP requests and all can be used to connect to our REST API.

Mobile apps are the primary potential consumers of our API. Because of their limited processing capabilities, they often rely heavily on servers' processing power. Using our API they can readily tap into it.

Any web browser can be used to communicate with our API through GET requests to retrieve and query data resources, and to even send commands. For more complex interactions (POST, PUT, UPDATE and DELETE) tools like Postman, curl and others can be used. These commands can be used to script the upload or download flat files to interact with legacy systems.

There are many applications and tools that can consume an OData API out-of-the-box. Notably, Microsoft Excel, Google Tableau, SAP Salesforce, etc.

SUCCESS STORIES

Besides exposing tables, (materialized) views, stored functions via GET requests and stored procedures via POST requests, our API has already been used to solve several integration problems. Note that POST, PUT, PATCH and DELETE requests are not normally performed on tables because of data integrity; we prefer to provide functions for these operations. Below are some real client challenges and our API solutions.

Challenge 1: Exposing Read-Only Data Maintained in Excel

Our client assembles specific data for custom orders in Excel files. When our WMS prints custom orders, it needs to retrieve specific information from the Excel files to include on the custom order for the workshop. The data contained in the Excel files was unique to that client and adding it to our WMS would have provided no value to other clients.

The Solution: We used our REST API to make the Excel data available to our WMS print order procedure. The client continues to maintain the custom order data in Excel and overwrites the Excel files exposed by the REST API whenever needed.

Challenge 2: Modifying Images

Product images are kept in our database. The database normally stores high-resolution images, but we have several applications in which low-resolution images are needed (a web catalog as as opposed to a printed catalog, for example). Other times, applications require square images or another specific shape.

The Solution: We created a set of API functions to resize, crop and/or modify the resolution of images on the fly.

Challenge 3: Updating Distributed Data

Our client wanted the ability to update customer records in both our POS and their third-party CRM. They initially planned to exchange customer info via CSV files daily. However, if a customer were to visit the client's store and connect to its website on the same day, customer info might change in both the POS and CRM. When customer records were exchanged, info may have been lost because the two systems would've overwritten each other.

The Solution: The REST API was used to exchange customer info in real time, removing any possibility of conflict.

Challenge 4: Posting a CSV File with Command-Line Tool

Our client wanted to export a list of all employees from its HRIS in a CSV file daily and send it to our Vision Store system.

The Solution: We created a specific POST endpoint in our REST API to accept the CSV file in the body of the request, parse it and then update to our employees table. The client used the curl command-line tool to post the CSV file.

CONCLUSION

Successful integrations using our REST API reinforces that the right design and development decisions were made based on years of integration experience.

Our API is drawing interest from existing Vision Suite clients and we anticipate that integration with future clients will be even faster and easier with enhanced performance.

JESTA I.S.

Enterprise Software for Retailers,
e-Tailers and Wholesalers

Let's Connect!

For more information on Jesta I.S.'s REST API and Vision Suite platform:

sales@jestais.com
www.jestais.com