Lets Build Inte-Greations with the Orchestrator!

Because why build good integrations when we can build great ones!

JD Edwards INFOCUS

Agenda

- **Intro**
- Integration Architecture Explained
- Great Integrations
- Why Orchestrator?
- Lets Go Build Some Inte-Greations

Hi 👋🏽 I’m Mo Shujaat!

VP of Advisory Services at ERP Suites

- VP of Advisory Services @ ERP Suites
- Over 12 years of JDE experience, across a diverse set of companies and industries.
- Distribution consultant by trade, orchestration builder by heart
- Experienced leading multiple JD Edwards implementations, upgrades, and process improvement projects for clients
- Enjoys building implementation/optimization roadmaps with clients and designing integrations

100+ Functional & Technical problem solvers

20+ yrs

Each consultant brings over 20 years of ERP experience

Advisory Practice

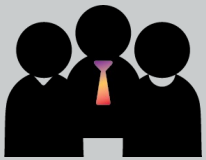
Integrating your ERP with emerging technologies to impact business goals.

Managed services

Extending your internal capabilities with first-rate response times.

Cloud services

Meeting demands for data accessibility with a seamless interface.



Consulting services

Applying expertise and best practices to maximize results.

erp suites

realize IT

ERP Suites Services

Business Advisory Services

- Project Management
- Technical Strategic Roadmapping
- Enterprise Architecture Strategy
- Systems Gap Analysis
- Process Engineering
- Organizational Change Management
- Digital Transformation
- Analytics & Insights Strategy

Functional Consulting Services

- JDE Distribution & Warehousing
- JDE Manufacturing
- JDE Financials
- JDE Human Capital Management
- Managed Services
- UXOne Expertise
- User Defined Objects
- Orchestration Design & Build

Technical & Infrastructure Services

- Technical Refresh
- Technical Upgrades
- Cloud Migrations
- IBM iSeries Administration
- Cloud Administration
- Networking & Server Infrastructure
- Identity Management
- Cybersecurity

Trusted Advisors

For JDE Users

unlocking efficiency | boosting profitability

ORACLE | Partner

erp suites
realize IT

The Right Partner for Managing Your JD Edwards System

ORACLE | Sell Partner

Expertise in
Oracle Cloud Platform
in North America

aws partner network
Select
Consulting Partner

25+
Re-platforms

125+
Cloud Migrations

175+
Upgrades

When we say we never fail, we mean it. We minimize disruptions and have **never** had to fall back on a project.

erp suites
realize IT

Agenda

- Intro
- **Integration Architecture Explained**
- Great Integrations
- Why Orchestrator?
- Lets Go Build Some Inte-Greations

Types of Integration Architecture

Point to Point

How it works: Applications are directly connected to each other through custom interfaces or APIs.
Pros: Simple to set up for a small number of systems, fast initial deployment.
Cons: Becomes unmanageable as the number of systems grows (spaghetti architecture)

Hub & Spoke

How it works: All applications connect to a central hub (integration broker or middleware) which manages routing, transformation, and orchestration.
Pros: Centralized governance, easier monitoring, reduces direct dependencies.
Cons: Hub can become a bottleneck or single point of failure if not designed for scalability.

Ent. Service Bus

How it works: A distributed messaging backbone connects applications via standardized protocols (often SOA-based).
Pros: Flexible, supports many integration patterns (sync/async, pub/sub, orchestration).
Cons: Can be complex to implement and maintain; heavyweight for modern cloud-native use cases.

API-Based

How it works: Applications expose functionality through APIs, which serve as the integration layer. Often structured as data access or business logic APIs
Pros: Promotes reuse, agility, and scalability; aligns with microservices and modern digital platforms.
Cons: Requires strong API governance, versioning, and security discipline.

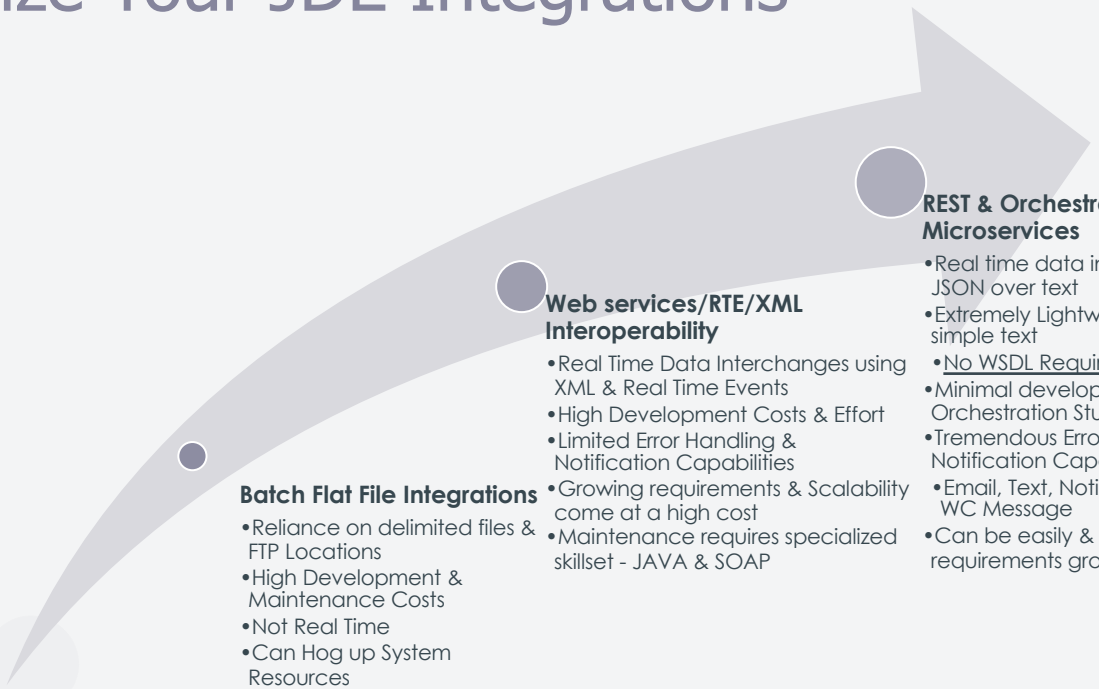
Event Driven

How it works: Applications communicate through events streams via a broker(e.g., Kafka, RabbitMQ). Consumers subscribe to events they care about.
Pros: Real-time, scalable, decoupled systems;
Cons: Harder to ensure transactionality and consistency; requires careful event design and monitoring.

iPaaS

How it works: Cloud-based platforms providing connections via various methods. Often combining APIs, DB connectors, flat files
Pros: lower infrastructure overhead, prebuilt connectors
Cons: Vendor lock-in, subscription cost, sometimes limited for highly complex, custom use cases.

Modernize Your JDE Integrations



Batch Flat File Integrations

- Reliance on delimited files & FTP Locations
- High Development & Maintenance Costs
- Not Real Time
- Can Hog up System Resources
- Non Existent Error Handling & Notification capabilities

Web services/RTE/XML Interoperability

- Real Time Data Interchanges using XML & Real Time Events
- High Development Costs & Effort
- Limited Error Handling & Notification Capabilities
- Growing requirements & Scalability come at a high cost
- Maintenance requires specialized skillset - JAVA & SOAP

REST & Orchestration Microservices

- Real time data interchanges using JSON over text
- Extremely Lightweight – JSON is just simple text
  - No WSDL Required!
- Minimal development efforts due to Orchestration Studio
- Tremendous Error Handling & Notification Capabilities
  - Email, Text, Notification, and even a WC Message
- Can be easily & quickly scaled as requirements grow

Agenda

- Intro
- Integration Architecture Explained
- **Great Integrations**
- Why Orchestrator?
- Lets Go Build Some Inte-Greations

Characteristics of a Great Integration

Timely: Gets data/runs processes/does stuff in the right place, at the right time

Consistent: Performs the same way every time (99.99%* of the time)

Communicative: Notifies appropriate stakeholders of events, exceptions, successes, and failure

Self-Sustaining: Integration should be able to operate on its own without human intervention

How to Inte-Great!

01

Qualify

- Qualify the type of integration that will work best for your scenario

02

Understand

- Understand how your integration will need to evolve & change over time

03

Place

- Ensure the integration fits into your overall systems integration strategy

04

Plan

- Plan your roadmap in advance
- Embrace MVP concept

05

Deliver

- Consider how your integration should execute its tasks
- Determine How stakeholders will interact with inputs & outputs

Agenda

- Intro
- Integration Architecture Explained
- Great Integrations
- **Why Orchestrator?**
- Lets Go Build Some Inte-Greations

Orchestrations

Once created are consumable API's via the AIS server

Log their state after each event with the AIS server

Allow Business analysis to create consumable microservices that can be used and reused

Can be daisy chained together to create even more powerful orchestrations

Connector Service Request

Consume External Rest API & Open APIs natively

Execute FTP/SFTP Protocols to transfer & receive files

Import & read .CSV files

Connect with the Local AIS service

Data Service Requests

Isolate data sets for processing

Retrieve data sets for transmission

Confirm data values

Chunk large data sets into Arrays for processing

Form Service Requests

Execute JDE form actions

Replicate user inputs

Bridge Integration to application

Return values from form & grid

Report Service Requests

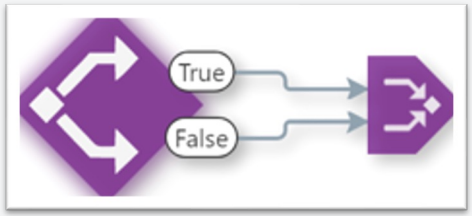
Execute JDE batch job actions

Replicate user batch job execution

Bridge Integration to application

Build Dynamic data selection & Processing option values

Rules



Build Light decision making into Orchestrations

Build additional validation and controls

IFTT Statements allow for conditional processing

Create divergent processing paths

Message Service Requests

Email notifications to a static or dynamically fetched email

Include relevant links or documentation

Attach JDE UBE Output as email attachment

Include links to call additional orchestrations for further processing

Logic Extensions

Create IF/ELSE logic, loops, variable assignments, and arithmetic operations

Do Table I/O, call business functions (BSFNs), or manipulate forms

Perform lookups, string functions, date math, and data transformations

Include links to call additional LEX, workflows, or orchestrations

Custom Service Requests

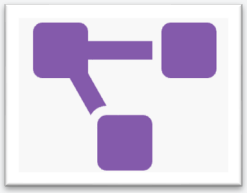
Use Groovy, Jruby, Jython to execute complex logic

Utilize Java Classes to execute custom logic

Utilize Standard & custom JDE business functions

Parse, Transform, or Build Arrays of data

Workflow



Build sequential or parallel approval chains that route orchestration data to users or roles

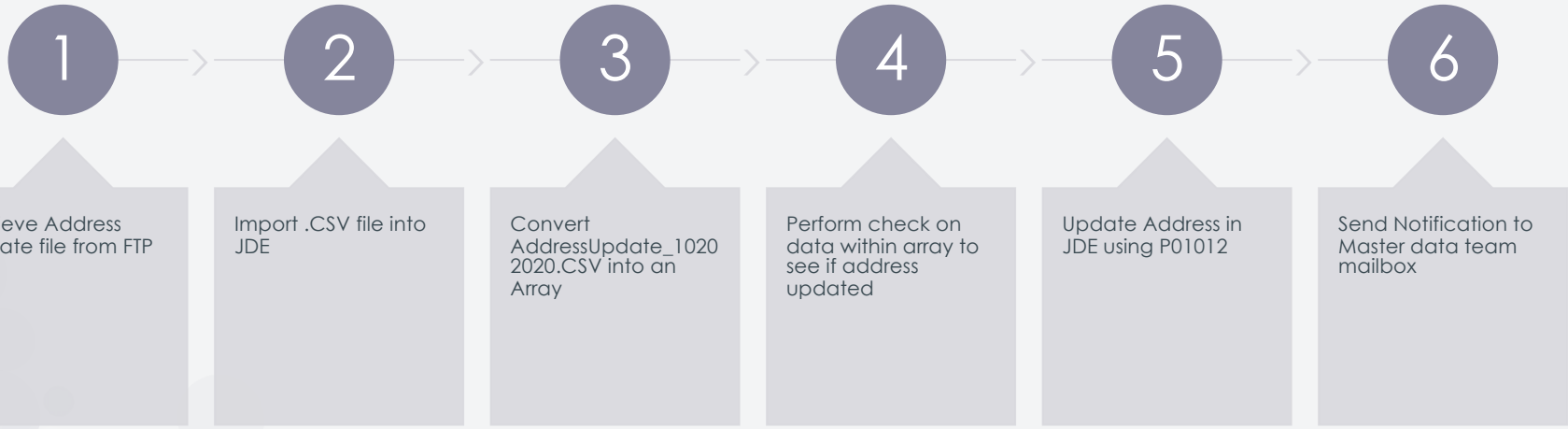
Trigger orchestrations (e.g., call REST API, update JDE tables, etc).

Send workflow notifications via email, or message center with contextual data

Configure escalations, reminders, and delegation rules

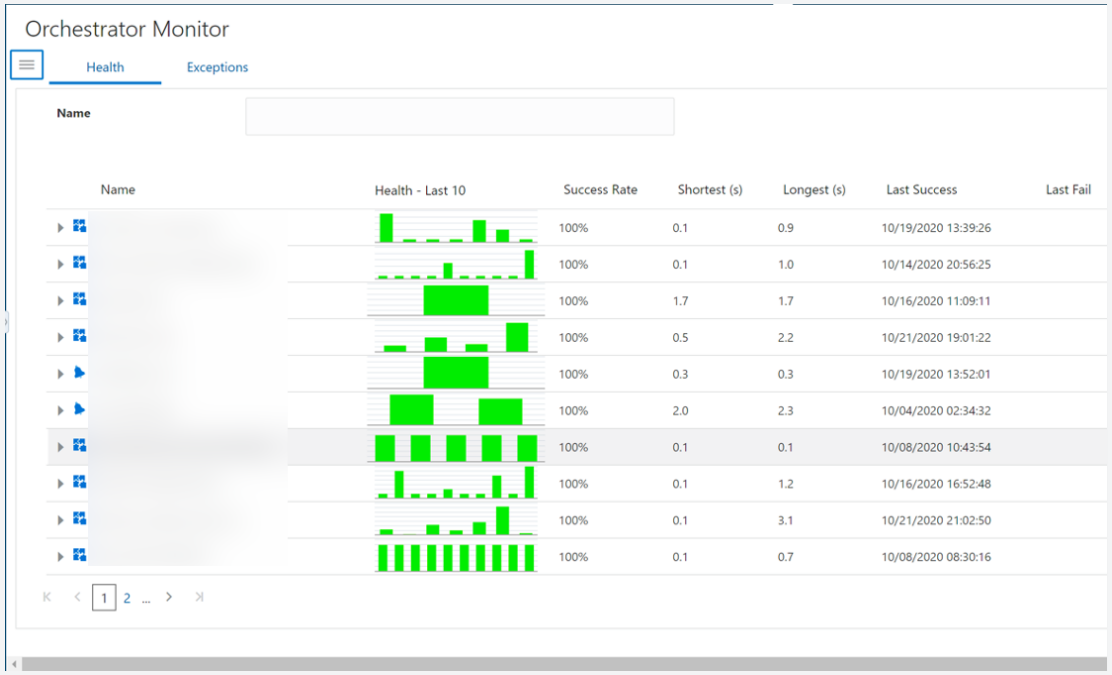
Secret to Orchestrators Power, Revealed!

Stringing together smaller re-useable services to perform large complex tasks with ease!



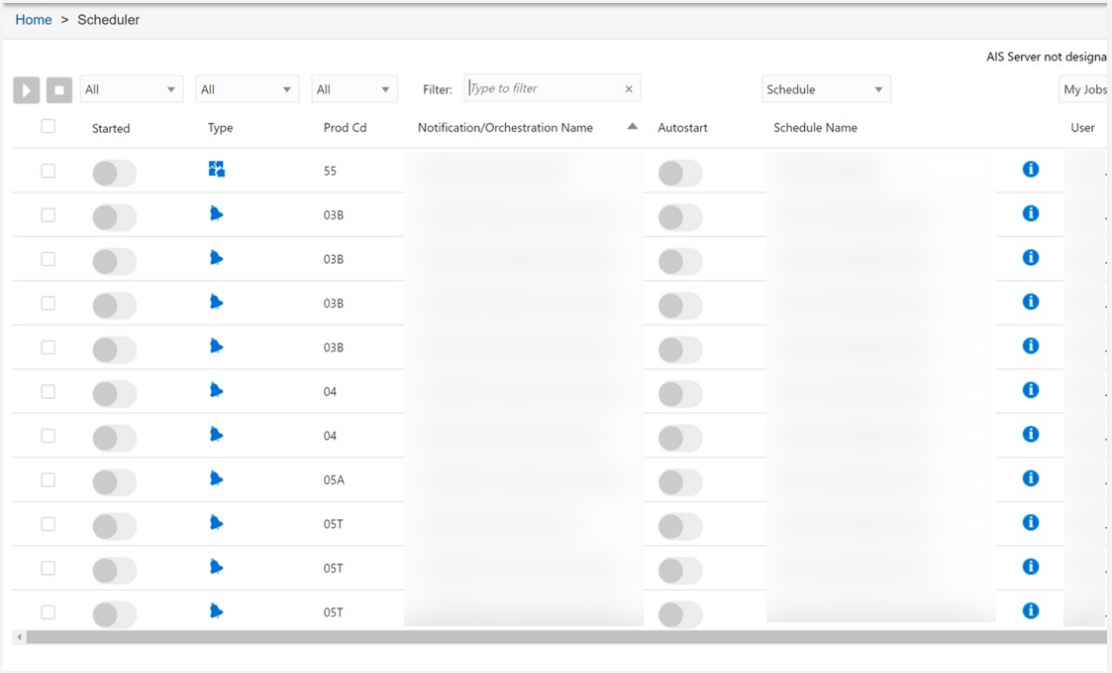
Orchestrator Monitor

- Tracks all orchestration activities
- Report orchestration runtime state
- Tracks & Logs orchestration exceptions
- Display orchestration runtime & execution data in reportable or graph like format
- Provides key IT/Orchestraton metrics

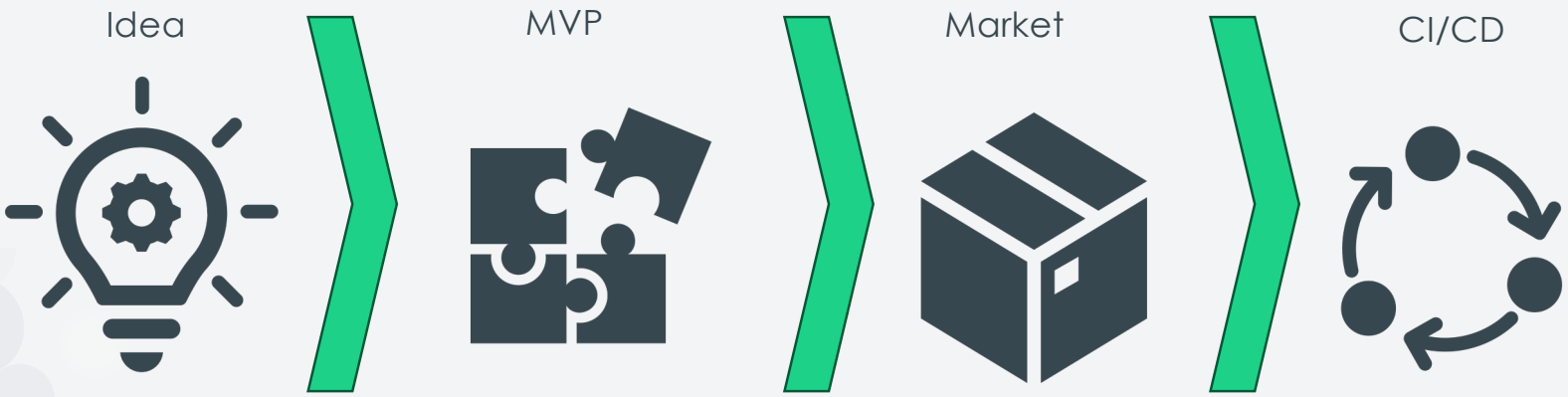


AIS Scheduler

- tracks all scheduled orchestrations
- Start or stop scheduled orchestrations
- Tracks & Logs scheduled run exceptions



The Most Important Reason Why…

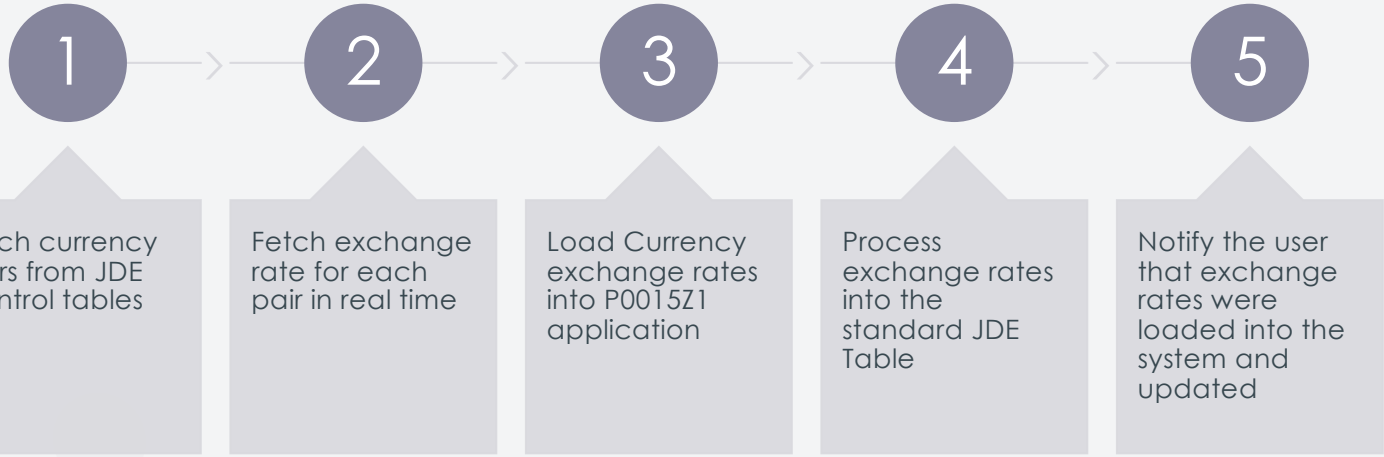


Agenda

- Intro
- Integration Architecture Explained
- Great Integrations
- Why Orchestrator?
- **Lets Build Some Inte-Greations**

Integration Use Case

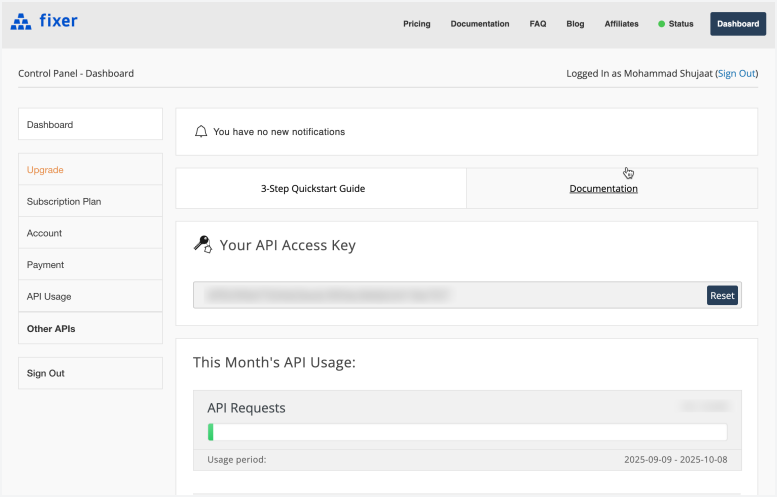
To fetch foreign currency exchange rates from a 3rd party system and update exchange rates in JDE Real Time



Get an API Key & Read the Documentation

Typically when doing any API based integration you will need to get authentication or a key so you can access the API – without this, you cannot consume the API

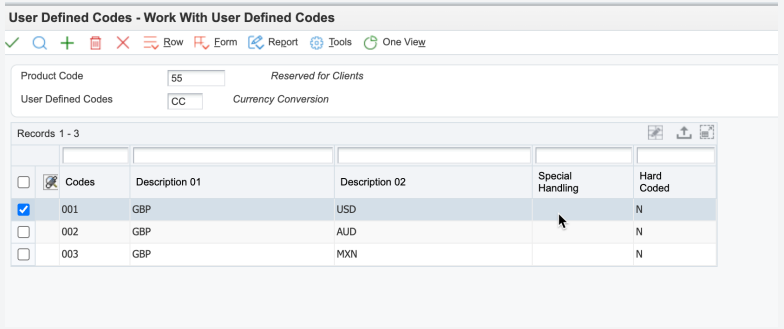
1. Go to fixer.io and sign up for a free account – this will give you access to your API Key
2. Save your API key somewhere secure - in this can it can always be retrieved from your account but this is not always the case!
3. Read through the API documentation to familiarize your self with the API structure and requirement
4. **Bonus: Use Postman to test the API before building an orchestration to get more familiar with its structure**



Store our “control” information

Many times an API will have to retrieve or fetch information from within JDE – make sure you are designing this in a scalable way so it can grow. Avoid Hardcoding

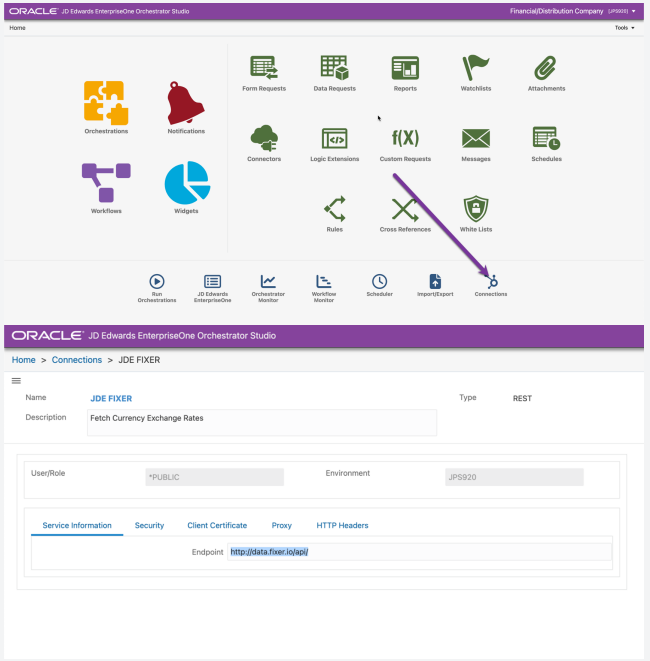
1. Log into JDE & navigate to the UDC application
2. Create a 55/CC UDC which will store your currency pairs
3. Load your currency pairs that you would like to retrieve and update on a daily basis into the UDC
4. **Note: You may not need to do this for all currency orchestrations but we will need to do it for this free API**



Build a Connec-TION

Every REST based API integration with orchestrator will require a connection (not a connector- that comes later)

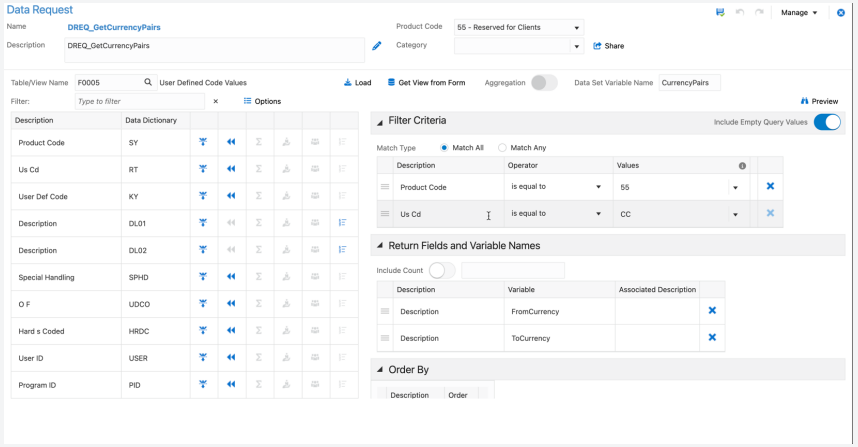
1. Login to Orchestration Studio in your environment and create a new connection
2. Connections utilize the JDE soft coded connections table F954001 and is environment and role specific
3. Connections are best created at the earliest common point of the API URL (i.e http://data.fixer.io/api/ vs http://data.fixer.io/api/latest)
4. **Note: Connections need to be recreated for each environment**
5. **Note: Remember you can always get more specific with your connections within the connector but its harder going the other way**



Build a Data Request

We will need to fetch our currency exchange rate pairs we stored in the UDC. These will need to be fetched as an Array

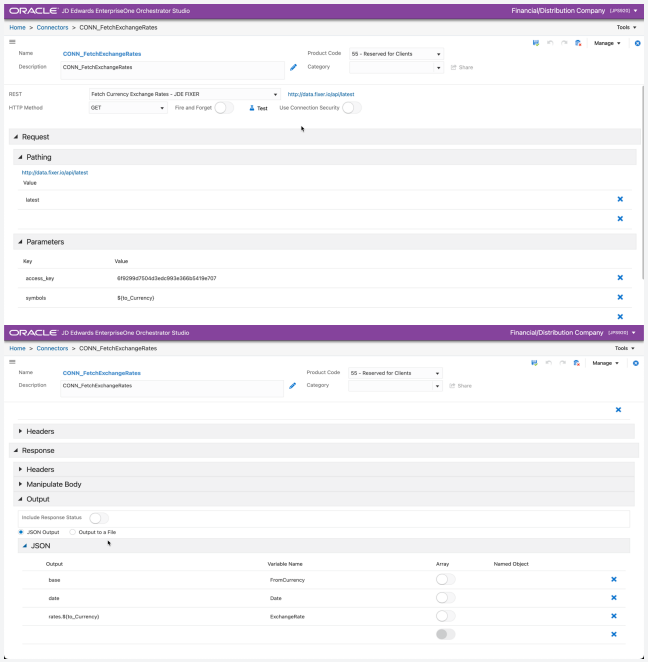
1. Navigate to Data Service Requests & Create a new data service request
2. Create your data request to fetch all the value from UDC 55/CC
3. Store the values from UDC 55/CC into a data set - give the data set variable name something easily recognizable
4. **Bonus: Debug your orchestration to see what your data service request is returning to test it early and often**



Build a Connec-TOR

Every REST based API integration with orchestrator will also require a connector service request – this is where we test our API

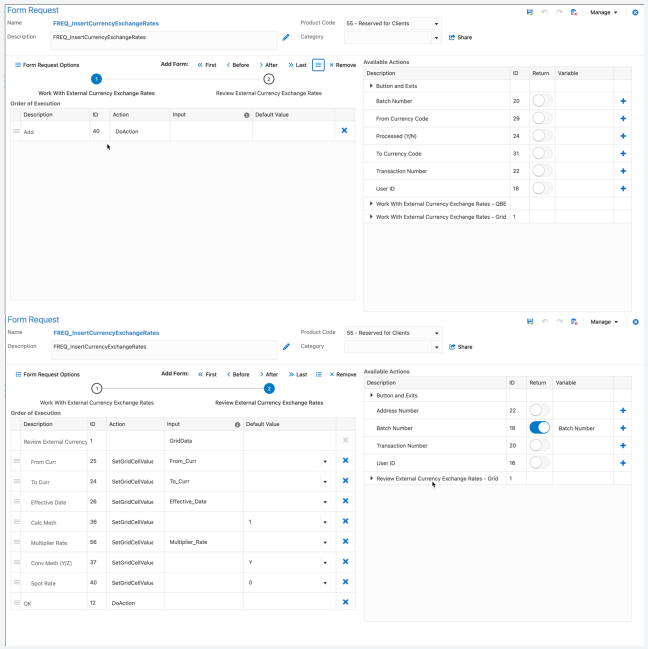
1. Navigate to Connector Service Requests & Create a new connector service request
2. Define your API Method & URL Pathing to get the URL to the exact endpoint we need (in our case http://data.fixer.io/api/latest)
3. Add any parameters, header values, body values (if POST)
4. Define any output manipulation logic via the connectors native capabilities of parsing or define custom logic via groovy
5. **Note: Keep a special look out for what goes into pathing vs parameters vs body's each are different!**



Build a Form Service Request

The form service request will allow us to enter data into JDE via a JDE application – this is a critical component of the orchestration

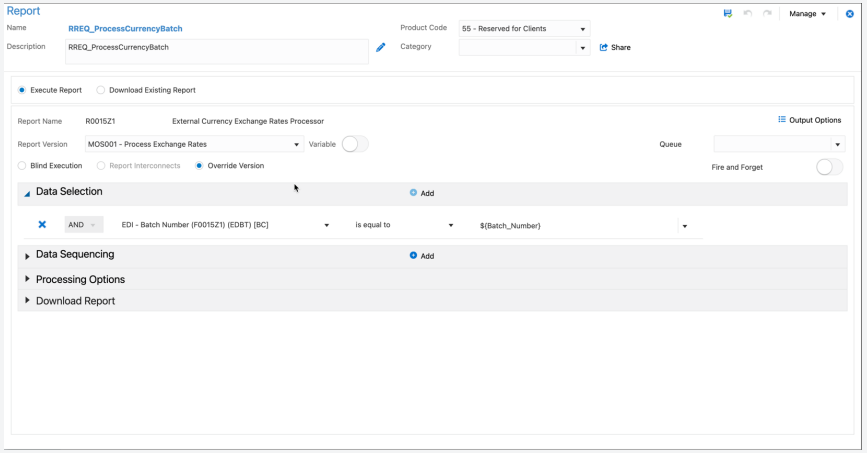
1. Navigate to the form service request within studio and create a new form service request
2. Add the P0015Z1 W0015Z1A and define its sole action – Clicking the Add button
3. Add the P0015Z1 W0015Z1C and define its sole action – Updating the grid with the currency values and clicking ok/select
4. Return the batch number from the form as an output to the form service request
5. **Note: You will need to declare variables for each of the values going into the grid**



Build a Report Service Request

The orchestration can call a UBE – the standard Z file processor to process the records in the Z table without having to write another form request

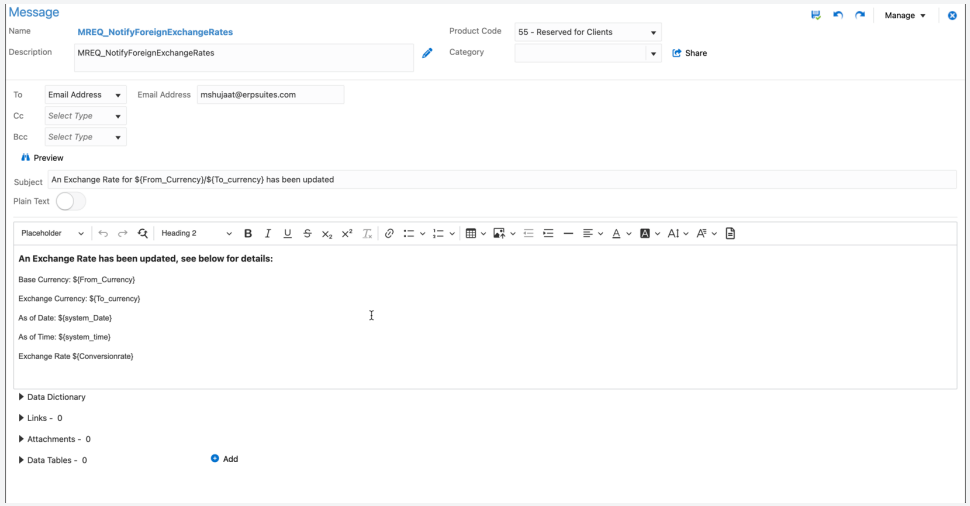
1. Navigate to the report service request within studio and create a new report service request
2. Select the R0015Z1 UBE and the version that you have created
3. Modify the data selection to pass in the Batch Number from the form request return variables
4. **Bonus: Add a form extension on the P0015Z1 application to kick off this UBE via a Button!**



Build a Message Service Request

A Great Integration should be communicative – so lets create a message to users or a group inbox notifying them of the exchange rate update

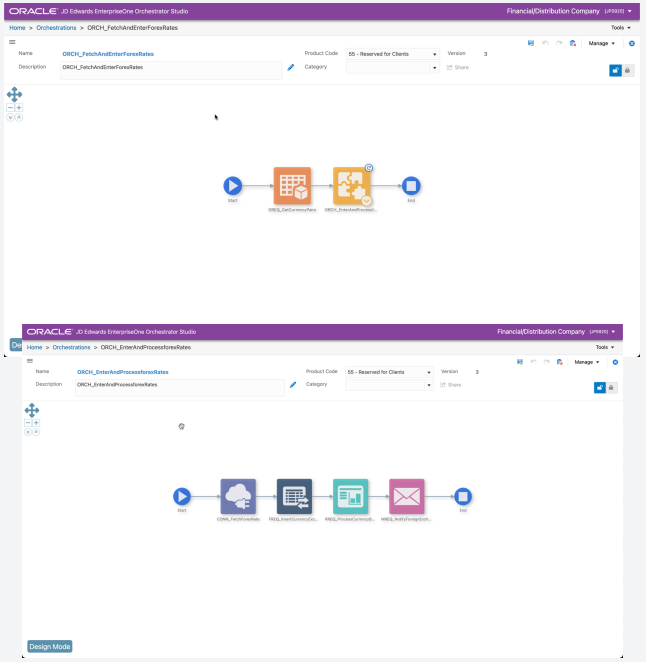
1. Navigate to the message service request within studio and create a new message service request
2. Define the recipients or recipient group
3. Define the message subject & body, using the variable convention where needed
4. **Bonus: Add a link to the currency exchange rate program that the user can click on to go straight to the exchange rate**

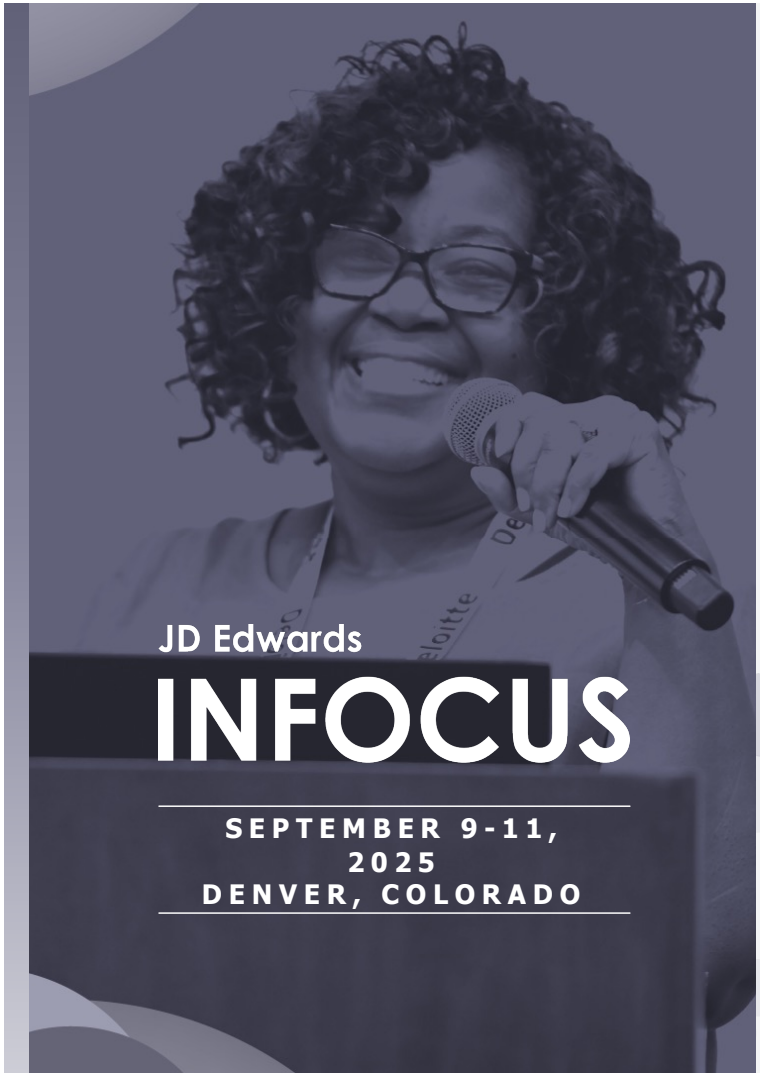


String It All Together & Test It

Add each component to the orchestration if you haven't been doing so all along, map the inputs/outputs, transformations, and then lets try it out!

1. Add the data request to its own orchestration with no inputs
2. Add the connector, form, report, and message requests to another orchestration with an input for "To_Currency"
3. Do transformations within Orch 1 make sure the called orchestration is iterated on the data set
4. Within Orch 2 make sure the transformations between each step are setup to pass info from each step to the next
5. Test your orchestration out!





Contact me:

Mo Shujaat: mshujaat@erpsuites.com

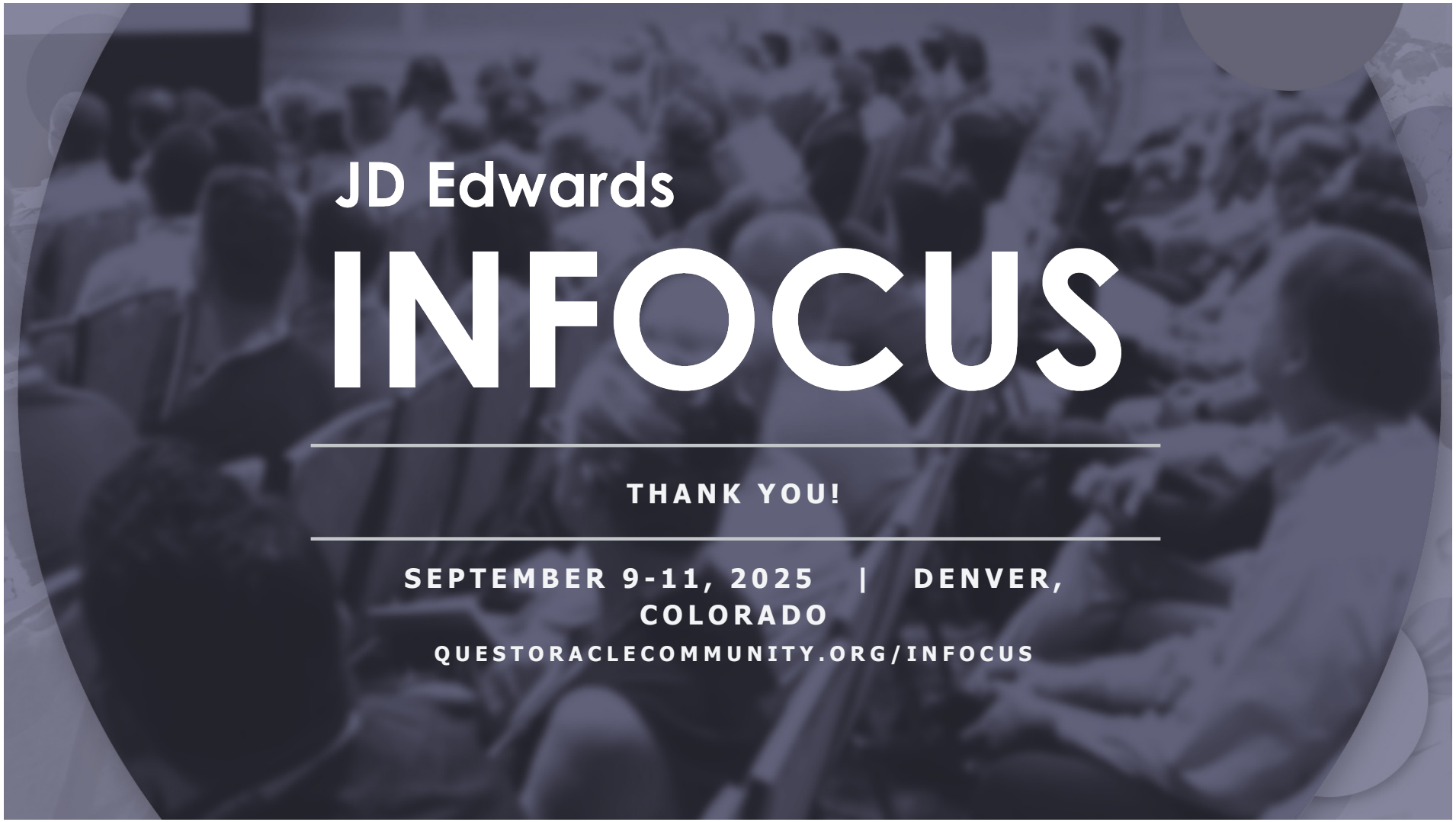
Session ID:

P-051299

Quest
JD Edwards
Community

LOVE THIS SESSION?

COMPLETE AT LEAST **3 SESSION SURVEYS** PER DAY AND GET ENTERED INTO THE DAILY DRAWINGS TO WIN A **$500** GIFT CARD!


JD Edwards
INFOCUS
THANK YOU!
SEPTEMBER 9-11, 2025 | DENVER, COLORADO
QUESTORACLECOMMUNITY.ORG/INFOCUS