Intro to AIS & Orchestrator (IoT)
Presented by Terry Dobbs

We make JD Edwards work better for people, and harder for business.
The AIS server provides the communication interface between JD Edwards and other integrations (mobile applications, web applications, java based programs, orchestrator, etc.).

AIS provides a “JSON over REST” interface to existing JD Edwards applications and forms using a standard HTML instance via APIs.

Installed much like an E1 HTML Server

Deployed via Server Manager to an Oracle WebLogic or IBM WebSphere server.
Using the AIS REST APIs you can perform actions such as:

- Setting field and QBE values,
- Press buttons or hyper items,
- Update, insert or delete grid records, select grid rows,
- Evaluate processing options, and interact with media objects.

With the Framework, you can perform virtually any action that a user using the standard application on a HTML client can perform!
You can send requests to the AIS Server using any REST client.

This example uses the tokenrequest API which is used to authenticate with E1. The endpoint URL is http://AISSERVER:PORT/jderest/tokenrequest.
REST APIs... Closer Look

JDE E1 API website contains in-depth information about all available APIs

About the REST APIs

Full list of E1 REST APIs available on the E1 REST API website https://docs.oracle.com/cd/E53430_01/EOTRS/
AIS Provides New Levels Of Integration
Orchestrator

Previously called Internet Of Things (IOT) Orchestrator

Orchestrator can be...

• An interface between IoT devices and E1 Applications
• Used to automate tedious, error-prone processes
• An integration platform to third-party services
• A notification engine
Orchestrations Are Microservices

Business analysts create microservices

- On-board Employee
- Alert Low Inventory
- Equipment Status Down
- Create Sales Order
- Alert Credit Limit

Application Interface Services

Orchestrator

APIs

Application Interface Services

Orchestrator Studio

JDE Apps

Business Logic

Business Data

JD Edwards EnterpriseOne
Orchestrator... Key Components

Orchestrations
Define the orchestration inputs and add service request, white list, rule, and cross reference steps.

White Lists
Define a list of authorized input values, for example a device's serial number. If the value is not in the white list, the orchestration terminates.

Rules
Define a set of conditions against which the input from the IoT devices is evaluated to produce a true or false status.

Cross References
Define relationships that map input values to JD Edwards EnterpriseOne values. For example, a device's serial number can be cross-referenced to an Asset Number.

Service Requests
Define the sequence of actions for invoking a particular process in JD Edwards EnterpriseOne applications or Java programs.

Types of Service Requests
- Service Request
- Form Request
- Custom
- Data Request
- Message
- Connector
- Watchlist
- Report
The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration.
Creating new form requests requires us to know how to accomplish the same task using the E1 web client. We can then use these same steps to build our request in Orchestrator Studio.
We need to load the E1 Application we want to interact with and choose the fields we want to interact with via our Orchestration.
Shown below are all of the steps in our new form request.

<table>
<thead>
<tr>
<th>Description</th>
<th>Mapped Value</th>
<th>Default Value</th>
<th>ID</th>
<th>Version</th>
<th>Form Mode</th>
<th>Return</th>
<th>Variable Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work With Addresses</td>
<td>P01072_W01072B</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone Numbers</td>
<td>P0115_W0115A</td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Order of Execution**

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Mapped Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Number</td>
<td>SetBEValue</td>
<td>AddressNumber_IN</td>
<td></td>
</tr>
<tr>
<td>Find</td>
<td>DoAction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select First Row</td>
<td>SelectRow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phones</td>
<td>DoAction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Action</th>
<th>Mapped Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Number for Update</td>
<td>UpdateRow</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Prefix</td>
<td>SetStrCellValue</td>
<td>PhonePrefix_IN</td>
<td></td>
</tr>
<tr>
<td>Phone Number</td>
<td>SetStrCellValue</td>
<td>PhoneNumber_IN</td>
<td></td>
</tr>
<tr>
<td>Phone Type</td>
<td>SetStrCellValue</td>
<td></td>
<td>HOM</td>
</tr>
<tr>
<td>OK</td>
<td>DoAction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We can use the controls in the top right-hand corner to save our request.
We can also define return fields that can be used in other parts of our Orchestration.

This request will be used in a rule to evaluate whether the address number being updated is the proper type. We only want to update “Employee” records.

Notice the highlighted “Return” column, here we can define the variable names we want to use to store the return values that we need.
Process Recorder

Enables us to record a series of actions in EnterpriseOne and save those actions as a new form request. This is an alternative to manually creating a form request in the Orchestrator Studio.

Invoked via standard E1 web client

Click Start and perform the actions you want to record

Click Stop to save your process, select the items you want returned to the orchestration
We can set different conditions we want to evaluate that will return either “True” or “False”. Based on the result of the rule we can tell our Orchestration what action to perform.
Message Service Requests

During an orchestration we can use the message service request to send messages to external email systems or the EnterpriseOne Work Center.

You will notice that we can pass in variables to our new message using the format $\{\text{Variable}\}$.

We also have the ability to add a JD Edwards shortcut to the bottom of our email.
You can configure a data request to query and return data from an EnterpriseOne table or business view or perform an aggregation of data from a table or business view and return aggregate amounts.

You don’t even need to know the specific view name, you can load it by specifying the application/form name.
Using connectors we can read/write to any non-JDEdwards database, call any RESTful endpoint, transfer files via FTP or even call another Orchestration.

**Defining a new connection**

**Adding a connector service request**
Report Service Requests

Report service requests give us the ability to trigger UBEs directly from our Orchestration.

Customize output options
We can utilize information from existing watchlists (critical/warning states, thresholds, counts) within our orchestration.
Now that we have created the necessary service requests, rule and message for our Orchestration we are ready to define our actual Orchestration.
New Orchestration... Continued

We need to select the service request we are adding to the Orchestration and then map input fields from the Orchestration to the required inputs of the service request.

Adding the Form Request we created to retrieve the Address Book search type

Mapping the input fields from the Orchestration to the added Form Request
We can add a second step by selecting the “Insert Step After” button. For the second step we want to add our previously created Rule.

Adding a second step after the Form Request for our Rule
Now we can add the final two steps. These will be dependent upon the result of our rule.

**Adding the form request to update phone number if Rule is “TRUE”**

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Iterate Over</th>
<th>Name</th>
<th>Service Request Input</th>
<th>Orchestration Input</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Request</td>
<td></td>
<td></td>
<td>Retrieve_ABSearchType</td>
<td>AddressNumber_IN</td>
<td>User Address Book Number</td>
<td></td>
</tr>
<tr>
<td>Rule</td>
<td></td>
<td></td>
<td>UpdateEmployee_Rule</td>
<td>PhonePrefix_IN</td>
<td>PhonePrefix_IN</td>
<td></td>
</tr>
<tr>
<td>Form Request</td>
<td>True</td>
<td></td>
<td>EmployeeUpdatePhoneNumber</td>
<td>PhoneNumber_IN</td>
<td>PhoneNumber_IN</td>
<td></td>
</tr>
</tbody>
</table>

**Adding the message service request to send the email if Rule is “TRUE”**

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Iterate Over</th>
<th>Name</th>
<th>Service Request Input</th>
<th>Orchestration Input</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Request</td>
<td></td>
<td></td>
<td>Retrieve_ABSearchType</td>
<td>${{Display(Name)}}</td>
<td>AlphaName.OUT</td>
<td></td>
</tr>
<tr>
<td>Rule</td>
<td></td>
<td></td>
<td>UpdateEmployee_Rule</td>
<td>$(PhonePrefix_IN)</td>
<td>PhonePrefix_IN</td>
<td></td>
</tr>
<tr>
<td>Form Request</td>
<td>True</td>
<td></td>
<td>EmployeeEmailUpdatePhoneNumber</td>
<td>$(PhoneNumber_IN)</td>
<td>PhoneNumber_IN</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>True</td>
<td></td>
<td>EmployeePhoneUpdate_Message</td>
<td>$(AddressNumber_IN)</td>
<td>User Address Book Number</td>
<td></td>
</tr>
</tbody>
</table>
New Orchestration... Continued

If we save the Orchestration and then back out to the Orchestration main window we can see a flow diagram of our Orchestration.
Now that we have our Orchestration we are ready to test. The AIS server provides an Orchestration Client we can use to perform tests. The client can be accessed using the address http://AIS_SERVER:PORT/jderest/client.
After entering our Input Values on the left and clicking Run we will see the Output on the right.

If the Orchestration is complex in nature it’s usually best to test out all the individual components first before running the Orchestration.
Our Orchestration ran successfully. You will see that the Phone Numbers application was updated in JD Edwards with the values we entered. We also received the E-Mail containing the information we defined in our Orchestrator Message.

Hi Demo User,

We just wanted to let you know that your E1 account has been updated.

Your new phone number is 777 888-9999. If this looks incorrect please let us know.

Phone Numbers
We can invoke our Orchestration directly through the AIS REST APIs using the Orchestration Service.

Example:

```
http://AISERVER:PORT/jderest/v2/orchestrator/EmployeeUpdate_Operation
```

It is very straightforward to execute an Orchestration directly via the Orchestration HTTP Endpoint.

Example:

```
http://AISERVER:PORT/jderest/v2/orchestrator/EmployeeUpdate_Operation
```
The Orchestrator Scheduler gives us the option to create and assign a schedule to an orchestration.

Creating a new schedule

Assigning schedule to an Orchestration
Notifications enable the system to notify users of specific business events as they happen without the need for the user to be online.
To receive notifications users must subscribe to them. Alternatively, in the latest Tools Release we also have the ability to assign notifications to users.
Orchestrator Notifications... Result

If a notification has been triggered (via the REST API or the Scheduler) and the parameters defined in the subscription are met then the user will be notified through the means they selected.
Orchestrator Licensing

- Orchestrator and Orchestrator Studio is now included in JD Edwards EnterpriseOne Core Tools and Infrastructure
- There is no license migration necessary, nothing to “buy”, just use it!
- EnterpriseOne applications invoked in orchestrations still need to be licensed
- “Application User” metric is for human users. “Connected Device” metric is for nonhuman users.
Questions?
Keep in touch.

erp-one.com