

## Middleware Connector

The CoupaLink certified and tested middleware solution will lower the total cost of implementation for your customer and make the implementation easier for them to support.

### Business Process Overview

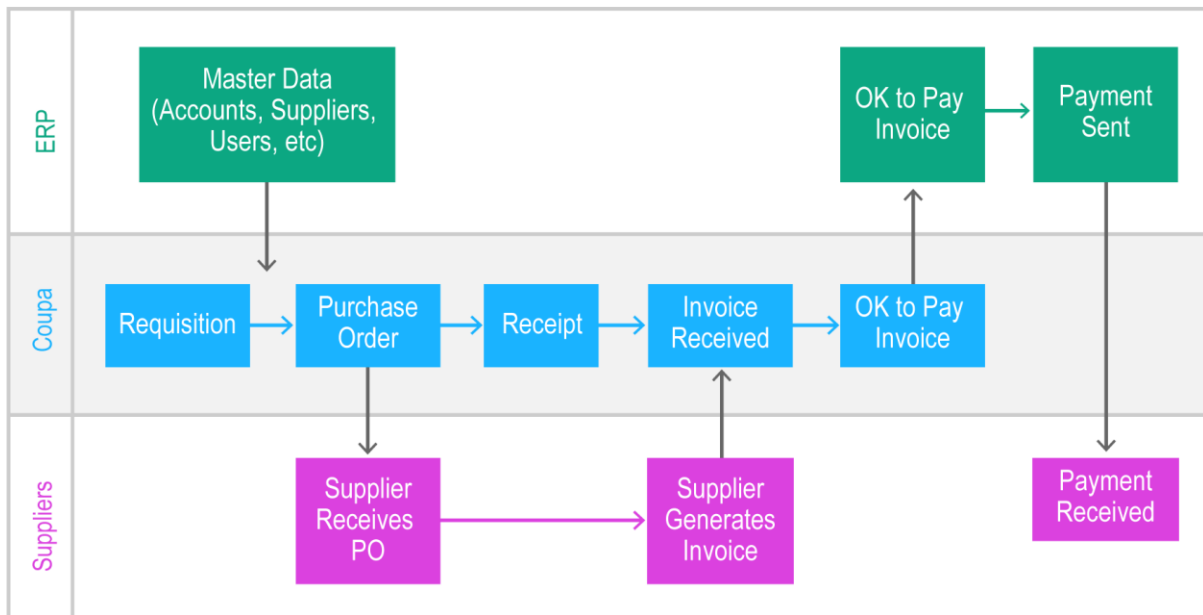
A middleware platform is used to simplify the process of connecting two or more systems with each other. A middleware sits in-between these systems and transforms data into a format that each respective system can process. In the context of Coupa, this typically means:

- Transforming the data coming into Coupa in a format that Coupa can process
- Transforming the data coming out of Coupa in a format the external system can process

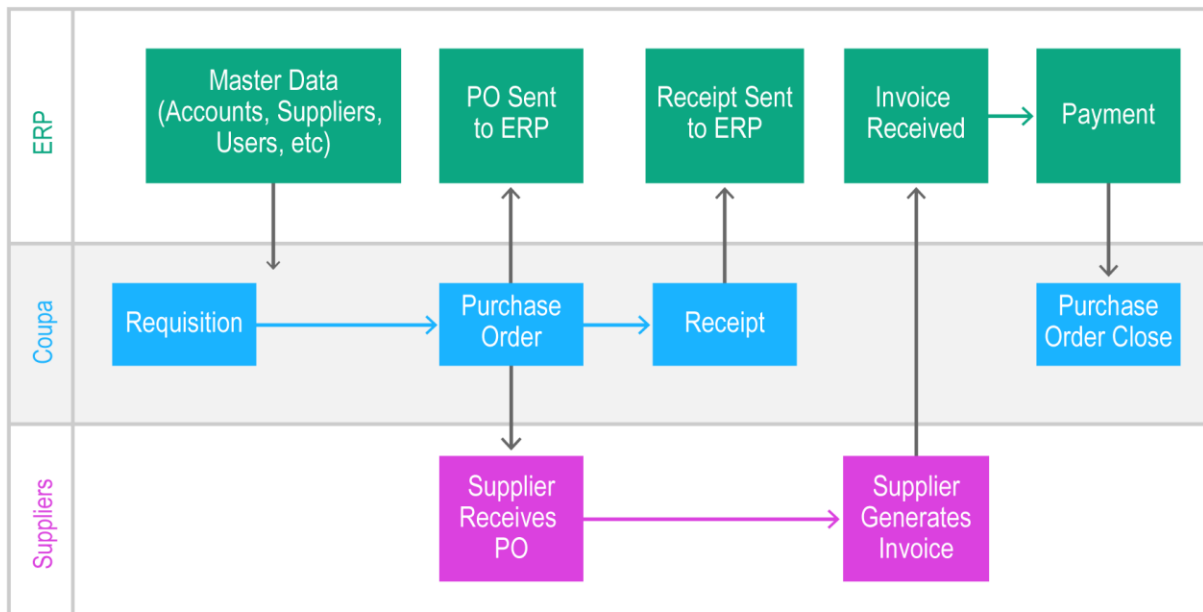
Organizations will use Coupa in various business processes. As you design your middleware solution, be mindful of these process workflows:

- Procure to Pay
- Procure to Order
- Expenses

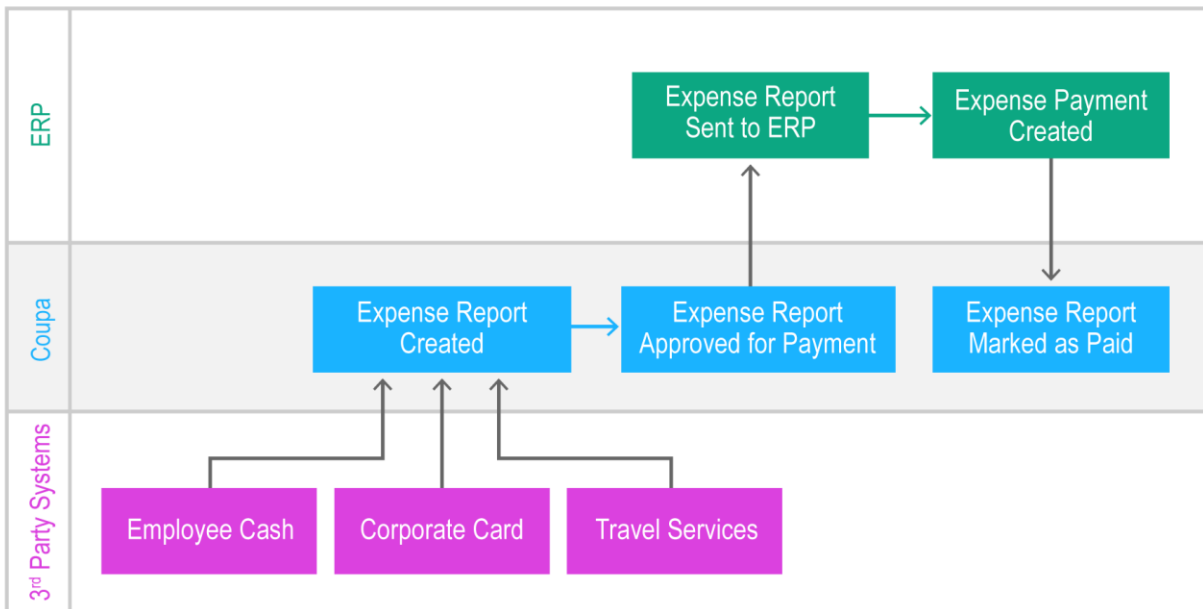
### Procure To Pay



## Procure To Order



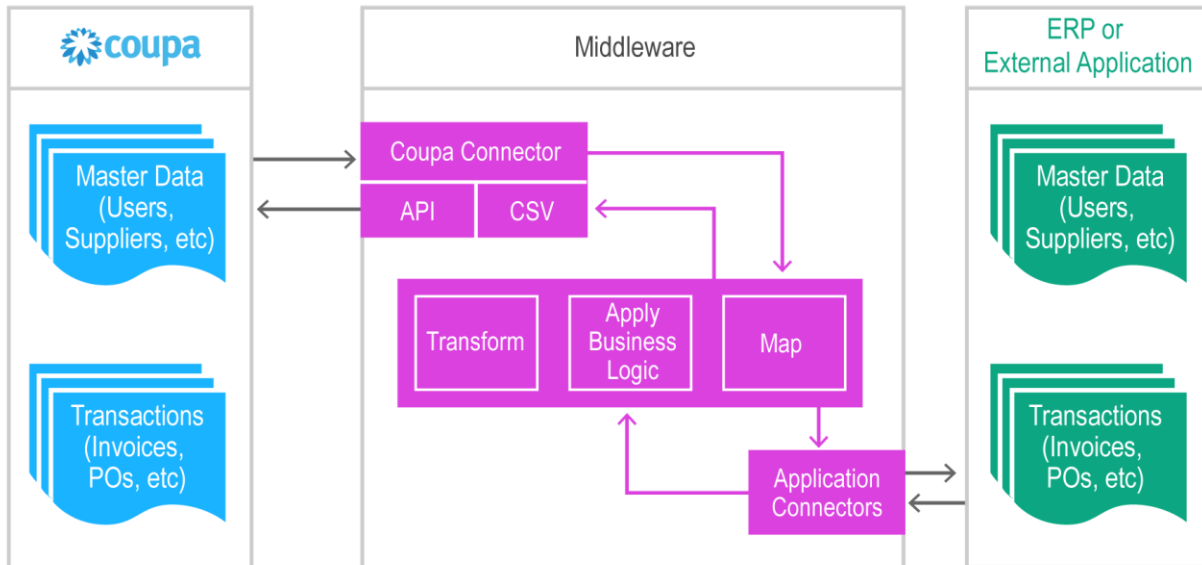
## Expenses





## Designing the Solution

Coupa has two main ways of integrating with other systems. We can either use flat file integration to send/receive data, or we can use our rest based APIs. Your middleware must be able to support Coupa API. Coupa Flat File integration support is optional.



Keep the following requirements in mind when designing your middleware solution:

### All Middleware solutions must support:

- Error handling
- Process multiple records at once with looping capabilities
- Monitoring capabilities
- Ability to “re-try” an execution
- Ability to map between two systems
- Ability to schedule an integration process (Can retrieve/create/update data from Coupa based on a Schedule – every 15 minutes, every hour, etc)
- Support Coupa Custom Fields (Retrieving & Updating Custom Fields)
- Persist data between integration executions. This is the ability to store data and make that data available for future integration runs. Examples:
  - Store last\_run timestamp. This gives you the ability to query for all objects in a system that were updated from the last time the integration ran.
  - Store last\_id processed
- (Optional) Change Data Capture – capture differences between two or more records
- (Optional) Embed custom code in integration flow. This gives you the ability to write custom code in the middle of a process to handle advanced mapping/transformation needs.



### Middleware solutions utilizing Coupa Flat File integrations must support:

- Ability to parse CSV files by column name and not position
- Ability to delete files going from Coupa to customer from the sFTP server
- Ability to create flat file CSVs with the following functionalities:
  - When configuring Column names in CSVs, the column names must be configurable instead of hard-coded
  - All fields are always text qualified (enclosed) with the double quote character: "
- New Line characters may be allowed within qualified field values.
- Be able to process CSV files with the following functionalities:
  - Double quote characters within a field can be escaped by another double quote character. If a user enters: Status is "Green" in a comment field, this would be represented as "Status is ""Green"""
  - Carriage returns and other special characters in text fields
  - Unlimited text lengths
  - Truncating long text values to target system length
  - Encode using UTF-8 (NO BOM)
- Ability to break files into parts if total file size is greater than 8MB

### Middleware solutions utilizing Coupa REST based API must:

- Not use strict XSD validation
  - You must support a “base schema” for each Object. If a user wants to add a new element (such as a custom field) to the base schema, you must provide a way for that user to do so.
  - Coupa does not have a valid metadata API, and does not provide schemas
- Support escaping/unescaping special xml characters
- Support arguments in the query
  - [https://success.coupa.com/Integrate/Technical\\_Documentation/API/Getting\\_Started/Arguments](https://success.coupa.com/Integrate/Technical_Documentation/API/Getting_Started/Arguments)
- Support paging in case the GET call exceeds 50 records
  - [https://success.coupa.com/Integrate/Technical\\_Documentation/API/Getting\\_Started/Querying\\_Options](https://success.coupa.com/Integrate/Technical_Documentation/API/Getting_Started/Querying_Options)
- Support proper encoding. For example, querying for invoices that have invoice\_number “Quality Assurance” would look like this:
  - [https://example.coupahost.com/api/invoices?invoice\\_number=Quality%20Assurance10](https://example.coupahost.com/api/invoices?invoice_number=Quality%20Assurance10)
- Support special actions:
  - [https://success.coupa.com/Integrate/Technical\\_Documentation/API/Getting\\_Started/Special\\_Actions\\_and\\_API\\_Notes](https://success.coupa.com/Integrate/Technical_Documentation/API/Getting_Started/Special_Actions_and_API_Notes)
- Support operators:
  - [https://success.coupa.com/Integrate/Technical\\_Documentation/API/Getting\\_Started/API\\_Operators](https://success.coupa.com/Integrate/Technical_Documentation/API/Getting_Started/API_Operators)

## Build your integration

### Flat File Documentation

[https://success.coupa.com/Integrate/Technical\\_Documentation/CSV](https://success.coupa.com/Integrate/Technical_Documentation/CSV)



## API Documentation

[https://success.coupa.com/Integrate/Technical\\_Documentation/API?](https://success.coupa.com/Integrate/Technical_Documentation/API?)

## List of Objects

Here is a table for all the Coupa objects for the CoupaLink certification program. This list consists of object name, direction of data flow (From / To), Channel (API, CSV or both) and whether support for the Object is (M)andatory or (O)ptional.

Object Name	CSV		API	
	From Coupa	To Coupa	From Coupa	To Coupa
Users	N/A	Yes (M)	Yes (M)	Yes (M)
Suppliers	N/A	Yes (M)	Yes (M)	Yes (M)
Remit To Addresses	N/A	Yes (M)	Yes (M)	Yes (M)
Supplier Sites	N/A	N/A	Yes (M)	Yes (M)
Supplier Business Groups (Content Groups)	N/A	N/A	Yes (O)	Yes (O)
Supplier Information	Yes (M)	N/A	Yes (M)	N/A
Lookup Values	N/A	Yes (M)	Yes (M)	Yes (M)
Accounts	N/A	Yes (O)	Yes (O)	Yes (O)
Account Validation Rules	N/A	Yes (O)	Yes (O)	Yes (O)
Addresses	N/A	Yes (M)	Yes (M)	Yes (M)
Approvals	N/A	Yes (O)	Yes (O)	Yes (O)
User Groups	N/A	Yes (O)	Yes (O)	Yes (O)
Business Groups (Content Groups)	N/A	Yes (O)	Yes (O)	Yes (O)
Attachments	N/A	N/A	Yes (O)	Yes (O)
Contract Terms	N/A	N/A	Yes (O)	Yes (O)

Legal Documents	N/A	N/A	Yes (O)	Yes (O)
Contracts	N/A	Yes (O)	Yes (O)	Yes (O)
Departments	N/A	Yes (O)	Yes (O)	Yes (O)
Data File Sources	N/A	N/A	Yes (O)	Yes (O)
Exchange Rates	N/A	Yes (M)	Yes (M)	Yes (M)
Integration Errors**	N/A	N/A	Yes (M)	Yes (M)
Integration History Records**	N/A	N/A	Yes (M)	Yes (M)
Integration Runs**	N/A	N/A	Yes (M)	Yes (M)
Inventory Transactions (Receipts)	Yes (M)	Yes (M)	Yes (M)	Yes (M)
Invoices	Yes (M)	Yes (O)	Yes (M)	Yes (PUT only) (M)
Supplier Items	N/A	Yes (O)	Yes (O)	Yes (O)
Items	N/A	Yes (O)	Yes (O)	Yes (O)
Budget Line	N/A	Yes (O)	Yes (O)	Yes (O)
Budget Line Adjustments	N/A	Yes (O)	Yes (O)	Yes (O)
Expense Reports	Yes (M)	N/A	Yes (M)	Yes (PUT only) (M)
Purchase Orders	Yes (M)	Yes (O)	Yes (M)	Yes (M)
User Addresses (Addresses)	N/A	N/A	Yes (O)	Yes (O)
User Business Groups (Content Groups)	N/A	N/A	Yes (O)	Yes (O)
Order Pads (or Order Lists)	N/A	N/A	Yes (O)	Yes (O)
Order Pad (or Order List) Lines	N/A	N/A	Yes (O)	Yes (O)



Payments (Invoice)	N/A	Yes (M)	Yes (M)	Yes (M)
Payments (Expense)	N/A	Yes (M)	Yes (M)	Yes (M)
Requisitions	N/A	Yes (O)	Yes (O)	Yes (O)

## \*\*Integration Errors, Integration Runs, Integration History Records

Integrations are becoming an increasingly important part of the Coupa platform. As customers get larger and use cases become more complex, there is a need to manage and monitor integrations closely, track errors, and notify customers when there are problems. Integration Run, Integration History Record, and Integration Error records allow you to perform this orchestration in the middleware using Coupa APIs.

Your middleware solution must be able to support the 3 different Integration Object API's. Before you can use the API's, you must make sure you have the ID of an existing *Integration* record. You must also make sure there is an *Integration Contact* set up in the Instance. An *Integration* is an Object that defines an integration (Outbound Purchase Orders, Inbound Invoices). An *Integration Contact* is a User who is notified via email whenever we want them to be notified using our Integration APIs.

### Sample usage of Integration Objects and the API:

Let's assume you have a scenario in which you want to query for all Approved Purchase Orders that have not been exported yet. After retrieving the POs and attempting to load them in an external system outside of Coupa, you may want to update Coupa to indicate whether or not the integration was successful. If the integration is not able to post to the external system, you then want to raise an alert and update the PO with the failure information.

The above scenario can be orchestrated using the Integration Errors, Integration Runs, and Integration History Record APIs. Below is a sample of steps you could take to address the scenario:

1. Go to <https://<instance>/integrations> to create an Integration record. The Integration can represent Outbound POs. Record the ID of the Integration (Example: **34**)
2. Go to [https://<instance>/integration\\_contacts](https://<instance>/integration_contacts) to create an Integration Contact. Set the alert-type to "Technical".
3. Create an Integration Run to have an Object that represents the PO Query. To create an Integration Run against the PO Query:
  - a. API POST [/api/integration\\_runs](https://<instance>/api/integration_runs)
  - b. Payload: `<integration-run><integration><id>34</id></integration><integration-run/>`
  - c. The response will return an Integration Run ID. In this sample, let's assume the Integration Run ID returned is **27**.
  - d. Set the Integration Run's "status" to "running" by calling [/api/integration\\_runs/<id>/run](https://<instance>/api/integration_runs/<id>/run)

4. Use the Purchase Order API to retrieve all POs that are issued and have not been exported yet: `/api/purchase_orders?status=issued&exported=false`
5. You will then want to loop through each PO to perform whatever processing you may need (such as sending it to an external system outside of Coupa). As each PO is processed and successfully posted to the ERP, you will want to create an Integration History in Coupa. You may also want to create an Integration Error in Coupa if that particular PO fails to be processed for whatever reason.
6. To create an Integration History Record against an Object and link it to an Integration Run.
  - a. API POST: `<instance>/api/integration_history_records/mark_exported`
  - b. Payload:
 

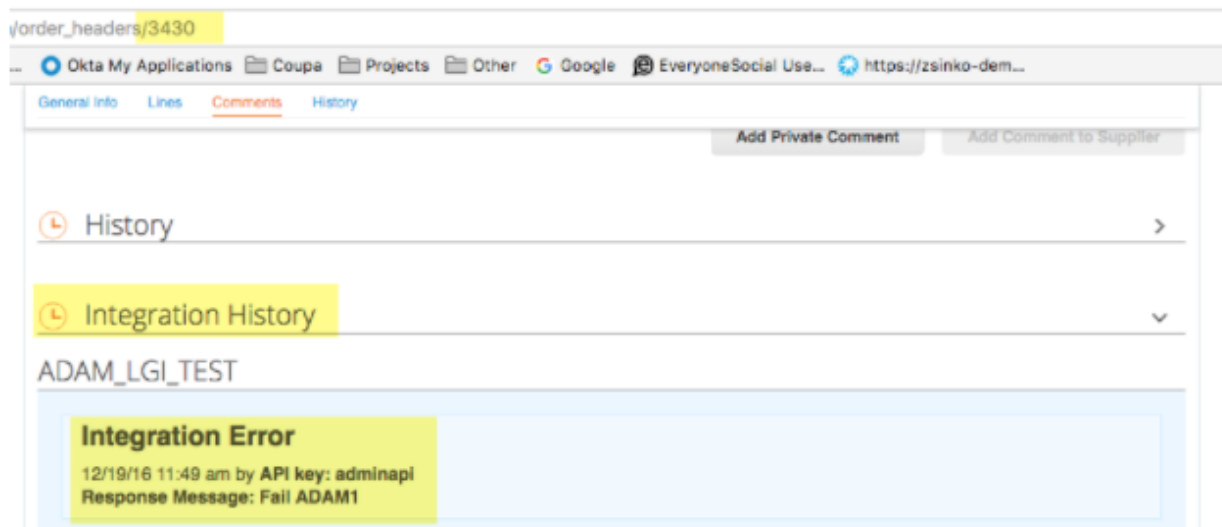
```
<?xml version="1.0" encoding="UTF-8"?>
<integration-history-record>
  <document-type>OrderHeader</document-type>
  <document-id>3430</document-id>
  <document-status>issued</document-status>
  <contact-alert-type>Technical</contact-alert-type>
  <responses type="array">
    <response>
      <response-code>Success</response-code>
      <response-message>IntegrationHistoryRecord Text</response-message>
    </response>
  </responses>
  <integration>
    <id>34</id>
  </integration>
  <integration-run>
    <id>27</id>
  </integration-run>
</integration-history-record>
```
7. To send an alert after creating an integration history record, follow the same steps as number 6, but use `"<instance>/api/integration_history_records/create_alert_and_mark_exported"` as the API POST path.
8. If the PO fails to be processed during the integration and you want to note that the Object failed to post in ERP, you need to create an Integration Error object in Coupa. To create an Integration Error object:
  - a. API POST: `<instance>/api/integration_errors`
  - b. Payload:
 

```
<?xml version="1.0" encoding="UTF-8"?>
<integration-error>
  <integration-run-id>35</integration-run-id>
  <contact-alert-type>Functional</contact-alert-type>
  <document-type>OrderHeader</document-type>
  <document-id>3430</document-id>
  <document-status>issued</document-status>
  <responses>
    <response>
      <response-code>Fail</response-code>
      <response-message>ADAM1</response-message>
    </response>
  </responses>
</integration-error>
```





In the above example, an integration error gets created for Purchase Order 3430:



9. Once all POs have been processed, close Integration Run as “success”. If there is an Integration Error, the status of the Integration Run will be “error”.

## Test Scenarios

### Middleware using Flat File Integrations:

1. Retrieve file from Coupa sFTP
2. Delete file from Coupa sFTP
3. Send file to Coupa sFTP

### Middleware using API Integrations:

1. Validate you are not using strict XSD validation
2. Process a single GET call that returns more than 50 records (support paging)

*For additional objects that are not listed below, please make sure to test Create/Update/Retrieve on those objects where applicable.*

### Object testing (for both API or Flat File integrations):

#### User Integration Scenarios (Optional)

- 1.1 Create a User
- 1.2 Modify a User
- 1.3 Update a Custom Field on a User
- 1.4 Deactivate a User
- 1.5 Reactivate a User
- 1.6 Add a Role to a User
- 1.7 Remove a Role from a User



1.8 Check Error Handling on Users - Update a user that does not exist in Coupa. Confirm error handling capabilities.

### **Supplier Integration Scenarios**

The Supplier object is a flat structure in Coupa as opposed to other ERP systems with parent-child relationships. This flattening task needs to be supported in the ERP Connector product.

Relevant associated data objects (Enterprise, Content Group, Commodity, Payment Term, Shipping Term, Tax Code) must exist as a prerequisite in Coupa to allow a successful automated integration.

- 2.1 Create a Supplier
- 2.2 Modify a Supplier
- 2.3 Create a Supplier with Multiple Locations/Purchase Org in ERP
- 2.4 Update a Custom Field on a Supplier
- 2.5 Deactivate a Supplier
- 2.6 Reactivate a Supplier
- 2.7 Change the Content Group on a Supplier
- 2.8 Change the Enterprise on a Supplier
- 2.9 Change the PO Method, PO Email on a Supplier
- 2.10 Check Error Handling on Suppliers

### **COA - Accounts / Lookup Integration Scenarios**

#### **Accounts**

- 3.1 Create an Account
- 3.2 Modify an Account
- 3.3 Deactivate an Account
- 3.4 Reactivate an Account
- 3.5 Check Error Handling on Accounts

#### **Lookup Values**

- 3.6 Create a Lookup Value
- 3.7 Modify a Lookup Value
- 3.8 Deactivate a Lookup Value
- 3.9 Reactivate a Lookup Value
- 3.10 Check Error Handling on Lookup Values

### **Approved Invoices Integration Scenarios**

- 4.1 Single-Line PO-Backed Invoice Approved and Ready to Pay in ERP
- 4.2 PO-Unbacked Invoice to ERP
- 4.3 PO-Backed Invoice with Custom Field to ERP
- 4.4 PO-Unbacked Invoice with Service Lines to ERP
- 4.5 PO-Backed Invoice with Multiple Tax Lines to ERP
- 4.6 PO-Unbacked Invoice with Attachments to ERP



- 4.7 PO-Backed Invoice/Credit Memo to ERP
- 4.8 Void an Invoice to ERP
- 4.9 50+ Approved Invoice and Ready to Pay in ERP (API Only)

### **Payment (Invoice) Integration Scenarios**

- 5.1 Create Payment in ERP and Update Coupa
- 5.2 Void Payment in ERP

### **Approved Expense Report Integration Scenarios**

- 6.1 Single-Line Expense Report Outbound
- 6.2 Split-Line Expense Report Outbound

### **Expense Payment Integration Scenarios**

- 7.1 Create an Expense Payment

### **Purchase Order Integration Scenarios**

- 8.1 Outbound PO - Single-Line PO
- 8.2 Outbound PO - PO with Multiple Line Items
- 8.3 Outbound PO - PO with Multiple Line Items and Split-Line Accounting
- 8.4 Outbound PO - PO Revision
- 8.5 Outbound PO - Single-Line PO with Service Item

### **Goods Receipt Integration Scenarios**

- 9.1 Send Created Goods Receipt to ERP
- 9.2 Send Void Goods Receipt to ERP

### **Budget Integration Scenarios**

- 10.1 Create a Budget Line
- 10.2 Adjust a Budget Amount
- 10.3 Update a Budget Line
- 10.4 Fetch Budget Line Query Options - Not applicable to CSV

### **Flat File CSV Parser Scenario**

- 11.1 Validate that you are parsing the CSV using column header and not column position. You can do this by opening a CSV and then swapping the first 2 header columns and associated values.



### Scenarios to walk through in Coupa Review Sessions

1. (Master Data) Create User with Custom Fields populated using middleware solution
2. (Master Data) Create Supplier using middleware solution
3. (Master Date) Create Chart of Account Lookup Values using middleware solution
4. Create multi line requisition with split billing in UI using master data above. Flip into PO
5. GET PO using middleware
  - a. Mark it as exported (API only)
6. Flip PO into Invoice using UI. Add tax amount either at header level or line level.
7. GET all invoices that are approved and have not been exported yet.
  - a. Update Invoice export flag (API Only).
8. Update invoice payment details
9. Retry execution – Retry User Import (step 1)
10. Change schedule for Supplier Import (step 2) to run hourly
11. Change schedule for Invoice Export (step 7) to run hourly
12. Download Logs from Middleware
13. Demonstrate error handling capabilities

### If supporting Coupa API's, you must also:

1. Create a new Integration Contact in the /integrations\_contact page
2. Create a new Integration Object in the /integrations page. This will be for Outbound Invoices
3. Use Integration Run API to represent the Invoice Query you will perform. Record the ID of the Integration Run.
4. Use the Invoice API to retrieve all invoices that have not been exported yet and are approved. (Please set up 2 invoices that fit these criteria before hand).
5. Loop through each Invoice within the Integration Run.
6. For the first invoice, create an Integration History Record against it and raise an Alert to yourself.
7. For the second invoice, create an Integration Error against it.

### Design Considerations

- Data Volume – How much data can flow between Coupa and your middleware platform? Does your solution support any Load Balancing technique?
- Staying up to date with Coupa Releases – Do you have automated test scripts to ensure your solution will work after each Coupa Release?
- Concurrent Job Scheduling – How can you both enable or disable concurrency?  
Example:
  - Concurrent Jobs Enabled: I may want to execute a Supplier Load and User Load into Coupa at the same time.
  - Concurrent Jobs Disabled: If I am exported invoices from Coupa every 15 minutes, I want to make sure the current Invoice Export job finishes before the next run. This is to avoid processing of any duplicates.