Oracle Warehouse Management Cloud

Integration API Guide

Release 26A

Oracle Warehouse Management Cloud
Integration API Guide

Release 26A

G47794-01

Author: Oracle WMS Cloud Product Team

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Access the online help from the user drop-down menu in the Warehouse Management application.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, and watch events.

Share Your Feedback

We welcome your feedback about Oracle Warehouse Managment. If you need clarification, or find an error, you can direct your questions via a service request to *My Oracle Support.*

2 Integration with Cloud WMS

Integration with Cloud WMS

Oracle Fusion Cloud Warehouse Management supports integration for the following categories:

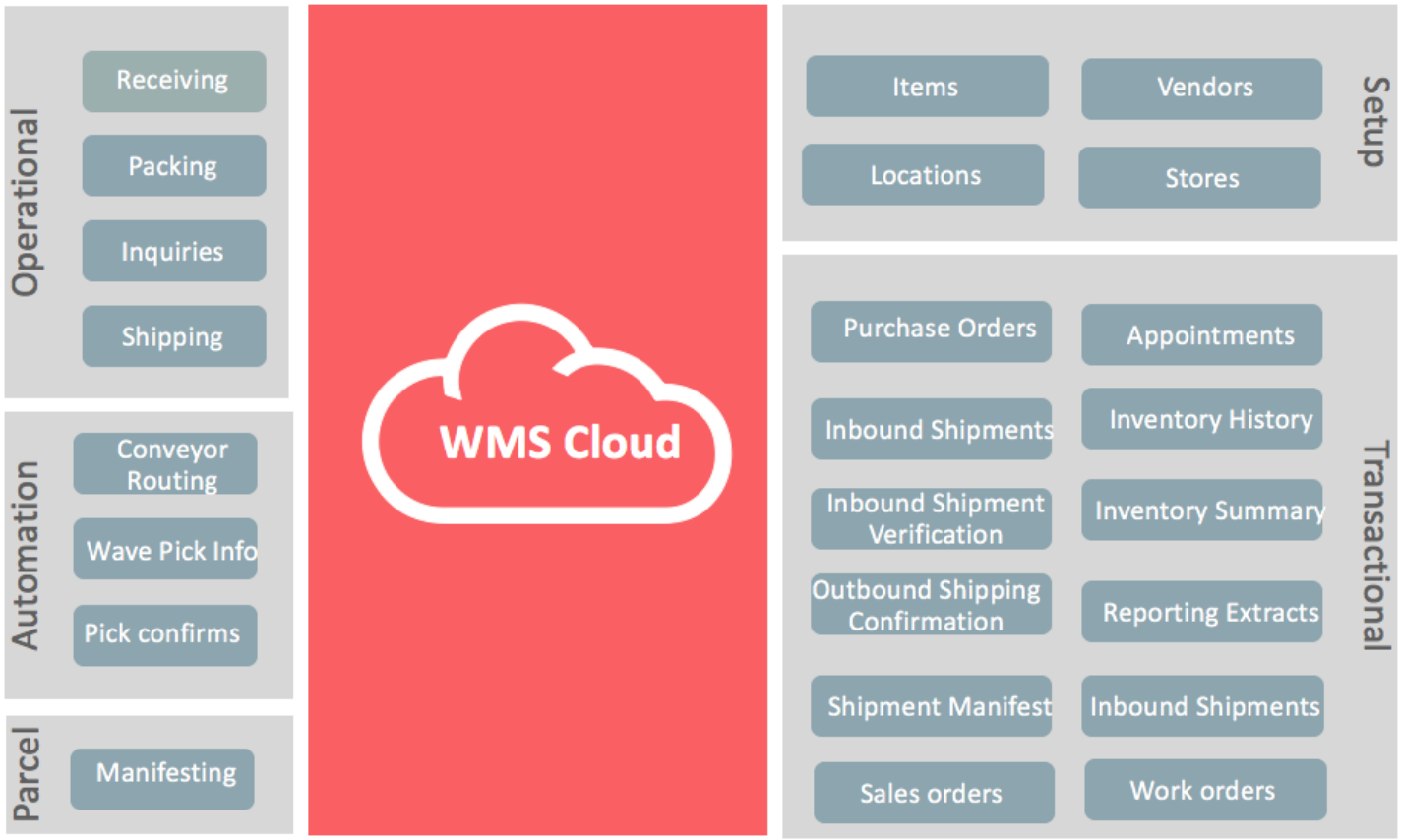
Automation and Operations

Parcel Carrier Integration

Setup and Transactional Data

Key Interfaces

The following flow describes the integration touchpoints:



Communication and Data

There are two main data formats supported by Oracle Warehouse Management (WMS) Cloud via interfaces and APIs:

- XML
- Delimited flat data

There are two main forms of communication protocols supported by Oracle WMS Cloud for integration with external systems:

- REST Web Service over HTTPS
- Secure FTP (SFTP) using an external SFTP site

Typically, the delimited flat data format is used over SFTP although it is also supported via certain WebServices. WebServices generally support an XML payload along with the delimited flat data format.

For Parcel integration, connection to external SOAP APIs is supported.

Note: WMS Cloud does not host SFTP sites. They must be hosted externally per the customers choice. WMS Cloud can transfer files to and from the external site.

Automation and Operations

A set of web service APIs are provided to handle the following:

- Integration with automated systems whether MHE or Voice
- To perform WMS operations that is invoked externally

Automation

Oracle WMS Cloud supports integration with MHE with:

- Standard Oracle WMS Cloud APIs

As of version 8.0.0 (2017), the prebuilt interfaces are supported, but will not be enhanced in the future. Standard Oracle WMS Cloud APIs will be enhanced to support all MHE operations. Unless explicitly noted, all incoming automation interfaces are only available via REST Webservices and not via SFTP/files.

Please see the *WMS Web Service APIs* for details.

Parcel Carrier Integration

Oracle WMS Cloud supports integration with several parcel carriers via multiple mechanisms:

- Integration with FedEx via Web Services provided by FedEx
- Integration with UPS via Web Services provided by UPS
- Integration with UPS, DHL GlobalMail via ConnectShip Web Services

For direct integration with FedEx and UPS, customers need to have an account with the carrier and obtain credentials to access the carrier web services from the Carrier. The customer then setups the Oracle WMS Cloud application with these credentials.

ConnectShip works similarly except that it is a third party that facilitates shipping via multiple carriers. Customers have to setup an account with Connectship in addition to the parcel carrier they wish to use.

As of version 20B, the above integrations will continue to be supported and enhanced.

Note: For more detailed information about parcel carriers, go to the *Oracle WMS Cloud Information Center.* From the top of the page, click on the **Documentation** tab at the top, then click the link under **Current Documentation** and refer to the **Parcel Carrier Integration Guide.**

Setup and Transactional Data

This section refers to list of setup and transactional entities that must be integrated with ERP or other 'host' systems in order to get data in and out of the WMS. Please refer to the *Interface Specfications* excel document for all the entity definitions and fields. The document is modeled towards the delimited file structure, however all the fields are the same as those used in the XML via Web Services. Refer to the XSD's for the XML schema definitions.

> **Note:** To access the latest XSD's, go to the *Oracle WMS Cloud Information Center* From the top of the page, click on the **Documentation** tab at the top, then click the link under **Current Documentation.**

There are three different ways to upload these entities into WMS:

- Upload excel or flat file via the Input Interface Screen in the application.

  > **Note:** To access the latest Interface Specifications, go to the *Oracle WMS Cloud Information Center* From the top of the page, click on the **Documentation** tab at the top, then click the link under **Current Documentation.**

- Oracle WMS Cloud provides excel templates and flat file format definitions.
- Use web services with XML payload to load the data.
- For XML, XSD schema definitions and sample XMLs are available
- Send flat files to the SFTP site (externally hosted site)

Touch point	Description
Items	Master item (SKU) definitions
Item barcode	Vendor barcodes
Item facility	Facility specific item properties
Item pre pack	Predefined kit definitions
Cubiscan	Item dimensional information
Vendors	Vendor definitions
Store	Destination stores for shipping
Location	Warehouse (DC) location defintions (bins)
Site	Facility stype of site. Used when shipping to a generic site or to model individual customers
Shipto Company	Destination companies
Asset	Assets such as high value pallets, totes used in a warehouse
Consolidation location map	Mapping stores and locations for store distribution configuration
LPN Location lock	Locate LPNs and lock. Used in initial data setup in combination with IB Shipment interface
Planned OB Load	Load planning for Orders from a TMS

Touch point	Description
Users	User definitions
Routes	Static routes definition
Price labels	Pricing information for Items
Purchase Orders	Purchase order transactional data
IB Shipments	Shipments of incoming inventory that may or may not be tied to PO's
IB Shipment serial nbr	Serial number information in IB Shipments
Appointments	Appointments for inbound loads
Orders	Sales orders (shipping requests)
Order instructions	Order picking/packing instructions
Work order	Work orders
Point of sale	Store POS update

Touch Point	**Description**
IB Shipment verification	Confirmation of received inventory
Inventory History	All WMS activities
Outbound Load shipment	Outbound shipment confirmation for LTL/TL
Parcel Manifest	Shipping confirmation for parcel
Inventory summary	Summary of all inventory
Wave Pick Info	Wave data to send to MHE or other systems

OAuth 2.0 Support for Integrations

Oracle Integration Cloud (OIC) is a complete and secure integration solution that enables you to connect your applications in the cloud. OIC simplifies the connectivity between your applications that live in the cloud and your applications that still live on premises.

OAuth 2.0 Support for Input Interfaces

Before you build an integration, you need to create a connection to the applications that provides minimal connectivity information for each system.

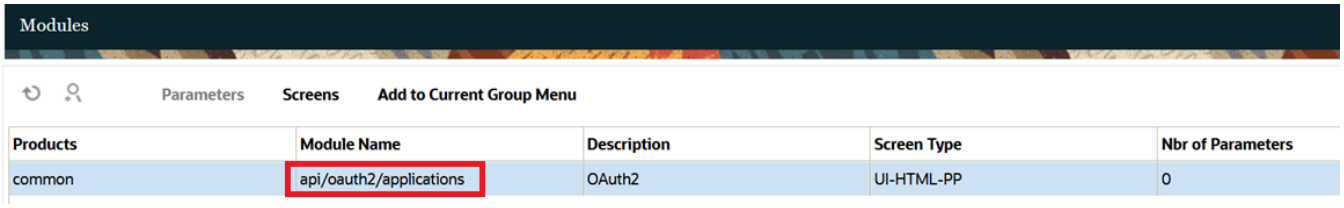
While setting up a connection in Oracle Integration Cloud (OIC) to connect to Warehouse Management (WMS) using the REST adapter, you can use one of the following security policies:

- Basic Authentication: For basic authentication, the username and password of the user in the WMS application are required.
- OAuth2.0: There is support in WMS to perform token-based authentication using OAuth2.0 with the following two grant types.
  - Resource Owner Password Credentials – You can log in to the application with the Password credentials.
  - Authorization Code – You will be redirected to the mentioned URL every time to get an authentication code.

Configuring the OAuth2.0 Authentication in WMS

Complete the following steps to configure OAuth2.0 in WMS:

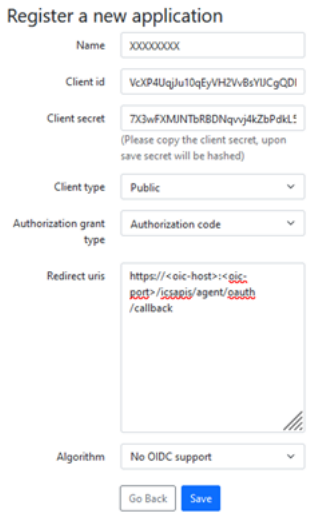
1. Create a screen using module **'api/oauth2/applications'** (Screen Type UI-HTML-PP).



2. Navigate to the created screen and click the **Click here** hyperlink to register a new application.

3. Use the created screen to configure client applications that will connect to WMS using OAuth2.0 to access REST APIs and click **Save**.

   Customers can have different applications to connect to integration tools, OIC is one of these.



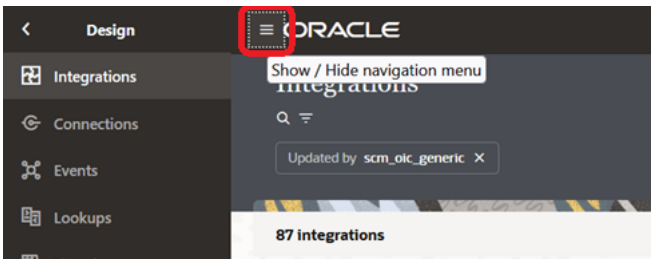
Note:

- A Client Id and a Client Secret will be generated for every application configured. If the authorization grant type selected is 'Authorization Code', then a redirect URI must be provided. For OIC, the Redirect URI will be – https://<oic-host>:<oic-port>/icsapis/agent/oauth/callback.
- If you are not an admin user in WMS, you will not be able to access the created screen even if the screen is assigned to your group's menu. You need to be part of a group that has been given permission 'OAuth2 / Manage OAuth2 Applications' to manage OAuth2.0 client applications.

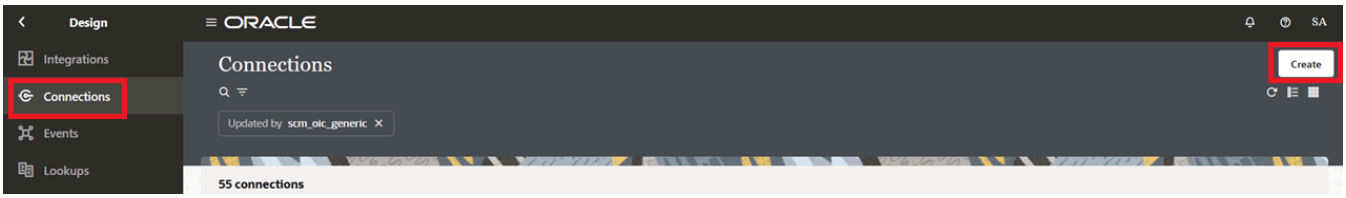
Configuring the REST Connection's Security Policy in OIC

For configuring a REST Connection Security Policy, do the following:

1. Log into the Oracle Integration application.
2. On the dashboard, click "View Integrations" button.
3. Click Show/Hide navigation menu icon.



4. Click **Connections** and then click **Create** button.

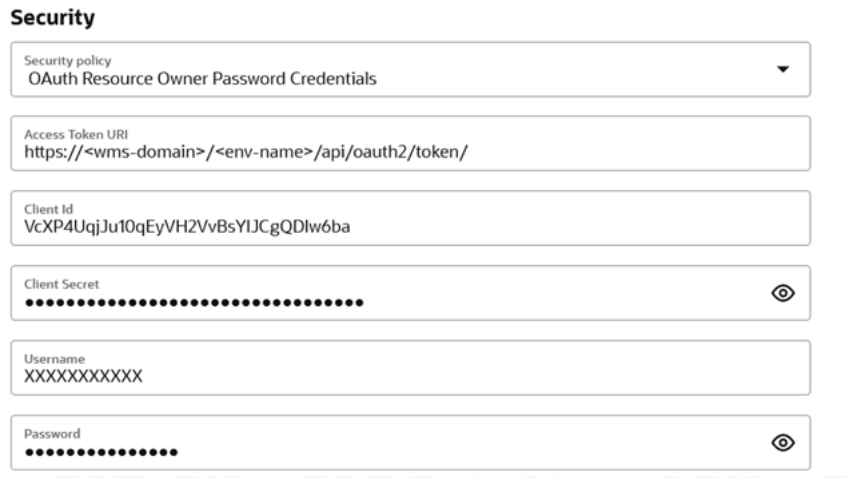


5. In the search box, search for REST Adapter connection and in the search results, click on the **REST**.
6. Enter the details and click **Create**.
7. On the connection screen, do the following and click **Save**:
    - Select a **Connection Type**.
    - Enter the **Connection URL** that corresponds to the selected Connection Type.
    - Select a Security Policy and enter the fields that pops-up based on the selected Security Policy.

    > **Note:** Based on the type of Security Policy selected in configuring a connection, the fields to enter will vary.

You can configure OAuth Resource Owner Password Credentials with the following fields:

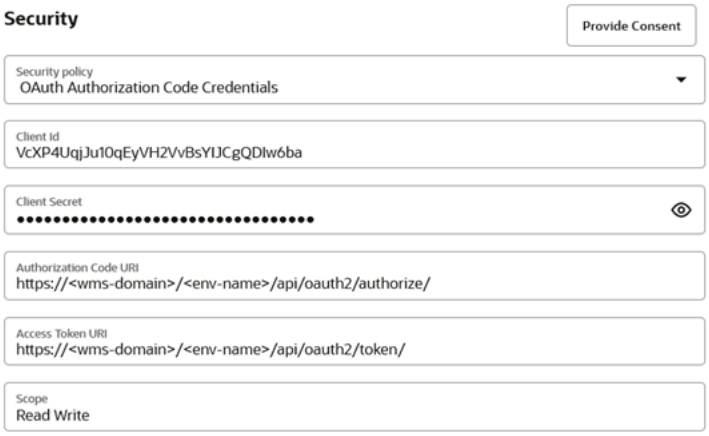
- Access Token URI: https://<wms-domain>/<env-name>/api/oauth2/token/
- Client Id: Client Id generated when configuring an application as described in in the above section.
- Client Secret: Client Secret generated when configuring an application as described in the above section.
- Username: Username of WMS user.
- Password: Password of WMS user.



You can configure OAuth Authorization Code Credentials with the following fields:

- Client Id: Client Id generated when configuring an application as described in the above section
- Client Secret: Client Secret generated when configuring an application as described in the above section
- Authorization Code URI: https://<wms-domain>/<env-name>/api/oauth2/authorize/
- Access Token URI: https://<wms-domain>/<env-name>/api/oauth2/token/

- Scope: Is not supported by WMS as of now. Enter “Read Write” as it is a required field.



You can configure the Custom Three-Legged flow with the following fields:

- Authorization Request: https://<wms-domain>/<env-name>/api/oauth2/authorize/?client_id=<client-id>&response_type=code&redirect_uri=https://<oic-host>:<oic-port>/icsapis/agent/oauth/callback
- Access Token Request: -X POST https://<wms-domain>/<env-name>/api/oauth2/token/ -H 'content-type: application/x-www-form-urlencoded' -d 'grant_type=authorization_code&code=${auth_code}&client_id=&redirect_uri=${redirect_uri}'

OAuth 2.0 Support for Output Interfaces

When Warehouse Management receives information, such as Purchase Orders, ASNs, etc., from external applications, the authentication protocols are supported using OAuth 2.0.

OAuth acts as an intermediary on behalf of you by providing the service with an access token that authorizes the information to be shared.

To allow you to configure, validate, and transfer data to external systems, the OAuth2.0 fields are available on the **Output Interface Configuration Detail** UI.

OAuth2.0 supported fields are available on the Create and Edit pane of the detail page for every interface type. The columns are hidden by default. You can add the fields to your UI view as needed.

For example, to add OAuth2.0 authentication protocol for one of the output interfaces, do the following:

1. Go to the **Output Interface Configuration** UI.

2. Select a record and click Output Interface Target (same as details icon) icon.

> **Note:** OAuth2.0 fields are not applicable for "Bill of Lading" and "Commercial Invoice" interface types, as e-mail and printer are the only two protocols supported "Bill of Lading" and "Commercial Invoice" interface types.

3. On the Create or Edit pane, Select **REST Web Service** as Interface Protocol, and add the following authentication fields to support OAuth2.0 Protocol.

Field Name	Field Type	Mandatory	Values	Comments
Interface Authentication Type ID	Drop-down	Yes	◦ Basic Auth ◦ OAuth2.0 - Client Credentials (CC) ◦ OAuth2.0 - Password	Drop-down to select the authentication type.
Client ID	Text Field	Conditional Yes		Mandatory if the grant type is one of the following: ◦ OAuth2.0 - Client Credentials ◦ OAuth2.0 - Password
Client Secret	Text Field	Conditional Yes		Mandatory if the grant type is one of the following: ◦ OAuth2.0 - Client Credentials ◦ OAuth2.0 - Password
Token URL	Text Field	Conditional Yes		Mandatory if the grant type is one of the following: ◦ OAuth2.0 - Client Credentials ◦ OAuth2.0 - Password
Scope	Text Field	Conditional Yes		Mandatory if the grant type is one of the following:

				◦ OAuth2.0 - Client Credentials ◦ OAuth2.0 - Password

3 WMS Web Service APIs

WMS Web Service APIs

Oracle WMS Cloud provides REST based Web Service APIs to perform various operations within the WMS. The currently available APIs are focused primarily towards data integration for getting data in and out of the application. A few additional APIs are available for key WMS operations. See the Technical Notes section for a detailed description of how Oracle WMS Cloud API request headers must be structured. The section also has some background information on APIs and Web Services in general. You can use the Chrome plugin Postman to try out accessing Oracle WMS Cloud web services. Make sure that the user you use has the permission listed below.

> **Note:** Any APIs not documented in the “REST API Guide” or the “Integration API Guide” books are meant for internal use by Oracle and subject to change without notice. We do not recommend you use them, but if you do, it is at your own risk. Such APIs may change or be dropped at any time, either mid-release or with the next release.

Authentication and Authorization

In order to access an endpoint, the request must contain, using BasicAuth, a valid WMS username and password. Within the WMS, the user must have the WMS permission "can_run_ws_stage_interface". The user must also have eligibility to any facility/company combinations represented in the data.

Response Structure

Oracle WMS Cloud API's respond with the following XML response (sample):

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <success>True</success>
 <response>
 <message>Process stage item submitted for file group 41415472</message>
 <errors/>
 <data/>
 </response>
</root>
```

Oracle WMS Cloud API's Error respond with the following XML response (sample) depending upon specific conditions in the message.

```
<?xml version="1.0" encoding="utf-8"?>
<root>
 <success>False</success>
 <response>
 <message>Record not found</message>
 </response>
</root>
```

Note: Earlier, the below listed API's returned HTTP status, **200 - OK** regardless of the success or failure of the API call. If API encountered an error condition, then the API returned 200-OK response with response body having **Success tag as "False"**. From 21D onwards, all the API's listed in the table are updated to return the HTTPS status, **400-Validation Error**, with response body having **Success tag as "False"**. This changes is implemented to align with Rest API Web Service principles.

The following is a list of commonly used response status codes:

Status Code	Status Message	HTTP Method	Description
200	Ok	HEAD, GET, POST	GET - The request was successful. HEAD - The resource exists. POST - Resource exists and/or has been modified.
201	Created	POST	Resource successfully created.
204	No Content	POST	The request was successful, but no content is being returned in the response body.
304	Not Modified	HEAD	The resource has not been updated since the target date-time.
400	Bad Request	HEAD, GET, POST	Invalid data or request structure.
401	Unauthorized	HEAD, GET, POST	Invalid login credentials.
403	Forbidden	HEAD, GET, POST	User lacks permission.
404	Not Found	HEAD, GET, POST	The resource does not exist.
405	Method Not Allowed	-	HTTP method is not supported for the requested resource.
409	Conflict	HEAD, GET, POST	Record Changed - The resource was modified by a concurrent operation before the request could be fulfilled. Try again.
500	Server Error	HEAD, GET, POST	An unhandled error occurred or the application was unable to formulate a valid response. Please contact support and provide any returned error information.

API	Description	Category	Initial Supported Version
Run Stage Interface	Trigger validation & processing of data already in stage tables.	Setup and transactional data	6.1
Update OBLPN Tracking Number	Update tracking number and other OBLPN parcel stats.	Automation & Operations	6.1

API	Description	Category	Initial Supported Version
Run MHE Stage Interface	Invoke an MHE interface to process data in MHE stage tables	Automation & Operations	6.2
Init Stage Interface	Main API for input data integration. Pass data in to validate and process	Setup and transactional data	6.4.0
Assign OBLPN to Load	Assign OBLPN to Load	Automation & Operations	7.0.0
Create LPN	API to create a single SKU IBLPN and associated inventory.	Automation & Operations	7.0.1
Update Output Interface	Set status and error message if any on a transmitted output interface	Setup and transactional data	8.0.0
Induct LPN	Induct LPN into MHE	Automation & Operations	8.0.0
Divert Confirm	LPN Divert confirmation from MHE	Automation & Operations	8.0.0
Load LPN	Load LPN	Automation & Operations	8.0.2
Entity Update API	Updates certain attributes of an entity	Setup and transactional data	8.0.2
From MHE Distribution Pack	From MHE Distribution Pack	Automation & Operations	8.0.2
From MHE Distribution Short	From MHE Distribution Short	Automation & Operations	8.0.2
Update Carrier LPN Label	Update Carrier LPN Label	Setup and transactional data	8.0.2

Deprecation of Legacy APIs

Warehouse Management Cloud is deprecating several legacy APIs that no longer comply with the overall strategy to develop REST APIs. The newer APIs ("lgfapi") provide improved functionality, structuring, stability, and scalability, and all customers must migrate to the newer, equivalent APIs. The time frame for deprecation is 22B, at which point the URLs that specify these APIs will no longer be accessible.

The legacy APIs to be deprecated are listed in the following table along with the equivalent lgfapi that you should migrate to. Only these legacy API's are being retired. Those not on the list will continue to be available for now, though they will eventually be retired as well, with sufficient notice.

API Name	Old API (to be deprecated)	New API (lgfapi equivalents)	Notes
Get Entity	GET wms/api/entity/<entity name>/	GET wms/lgfapi/v10/entity/<entity name>	
Get Entity Status	POST wms/api/get_status/<entity name>/	GET wms/lgfapi/v10/entity/<entity name>	
Get Extended Property	GET wms/api/extended_property/<entity name>/	GET wms/lgfapi/v10/entity/<entity name>	See lgfapi documentation for additional functionality

API Name	Old API (to be deprecated)	New API (lgfapi equivalents)	Notes
Get Seq Number	POST wms/api/get_next_numbers/	POST wms/lgfapi/v10/entity/seq_counter/get_next_number	
Lock/Unlock LPN	POST wms/api/lock_unlock_lpn/	POST wms/lgfapi/v10/entity/container/lock POST wms/lgfapi/v10/entity/container/unlock	Also supported via "iblpn" and "oblpn" entities
Pick Confirm	POST wms/api/pick_confirm/	POST wms/lgfapi/v10/pick_pack/pick_confirm	
Receive LPN	POST wms/api/receive_lpn/	POST wms/lgfapi/v10/entity/iblpn/receive	
Ship OBLPN	POST wms/api/ship_oblpn/	POST wms/lgfapi/v10/entity/oblpn/ship	
Update Active Inventory	PATCH wms/api/entity/active_inventory/<key>/	POST wms/lgfapi/v10/entity/location/update_active_inventory	
Update OBLPN Dimensions	POST wms/api/update_oblpn_dims/	PATCH wms/lgfapi/v10/entity/oblpn	

Run Stage Interface

Example URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/run_stage_interface/"

Initial WMS Version: 6.1

Overview

API to run the data import process for a stage table. It works on data already in the staging table. To load and process data, please use the InitStageInterfaceAPI.

- If no file_group_nbr is supplied, then the interface is run for every record in status Ready for the Company/Facility.
- If file_group_nbr is supplied, then all records for that group in status Ready for the Company/Facility are processed. Note that not all interfaces support the use of file_group_nbr.

> **Note:** This API is not meant to upload data and process. It's only meant to process data that has been loaded through other means. For uploading and processing data, please refer to the "Init Stage Interface" API.

Update OBLPN Tracking Number

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/update_oblpn_tracking_nbr/"

Initial WMS Version: 6.1

Overview

API to update an OBLPN's tracking number and optionally weight and/or ship via.

If a Carrier LPN record for the OBLPN does not exist, the API will attempt to create it. The ship via from the allocated order will be used if one is not supplied in the arguments.

Requirements

1. The LPN must exist for the facility/company provided
2. The LPN must be of type outbound
3. The LPN status cannot be CANCELLED
4. The ship via must exist for the same company as the LPN

Request Arguments

Argument Name	Function	Required	Data Type
company_code	WMS Company Code	X	string
facility_code	WMS Facility Code	X	string
oblpn_nbr	Container number to be updated	X	string
tracking_nbr	Carrier tracking number	X	string
ship_via_code	Updated ship via		string
weight	Updated weight		decimal
rate	Update rate		string
master_tracking_nbr	2nd tracking number		string
estimated_delivery_time	Estimated delivery date (yyyymmdd)		string
dry_ice_weight	Updated Dry Ice Weight		decimal
label	Image of label		Base64.PDF
carrier_webservice_label_type	Webservice label type		string

Note:

- Since version 9.0.0, new company parameter max_allowed_wt_vol_dim_decimal_scale controls the decimal precision for the following fields: weight and dry_ice_weight
- The precision of rate is set at 2

Carrier Webservice Label Type

The “carrier_webservice_label_type” parameter allows you to specity the web service label type that will be uploaded. This field accepts the following values:

- ZPL
- PDF
- IMAGE

For example: If you are uploading the ZPL code for the Carrier LPN Label, then the label type should be set to ZPL.

If the label is sent without sending the label type, the existing API behavior is unchanged. However, if the label type is sent without the label, the API will send an error with the message "label type without label is disallowed.”

Run MHE Stage Interface

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/run_mhe_stage_interface/"

Initial WMS Version: 6.2

API to run the data import process for a staged MHE data or to run a custom MHE function.

Requires Celery to be enabled as all messages are run asynchronously by default.

Request Arguments

Argument Name	Function	Required	Data Type
entity	Interface name	X	string
company_code	WMS Company Code	X	string
facility_code	WMS Facility Code	X	string
mhe_vendor_code	MHE Vendor	X	string
python_function	Name of custom MHE function to be run. Required if entity is 'custom'.		string

Supported Entities

Entity Value	Interface Name	Initial WMS Version
from_mhe_lpn_diverts	LPN Divert Confirmation	6.2
custom	Custom MHE Function	6.2

Init Stage Interface

Example URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/init_stage_interface/"

Overview

API to insert data into stage table(s) and run the interface to process the data.

Note: This is the API to be used by external systems to exchange setup and transactional data with the WMS.

All information regarding the interface entity and file groupings are within the XML in the <Header> tag.

This API supports XML format for all supported entities and flat format for a subset.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type	Initial Supported Version
xml_data	XML to be processed	C		string	-
async	Run stage table entity asynchronously		True	boolean	-
validate_xml	Throws hard error on XML structure or data issues		False	boolean	7.0.1
flat_data	Pipe-delimited data to be processed	C	''	string	8.0.0
entity	Type of flat data being interfaced. Only used in conjunction with flat_data.	C	''	string	8.0.0

Assumptions

- Either xml_data or flat_data must be passed - not both.
- "async" is applicable for both XML and flat for the processing of stage data

Flat File Data

- Mimics the same process as if you had uploaded it via Input Interface UI
- API keys 'flat_data' and 'entity' are required
- Only entities "item" and "order'" are currently supported
- Supported Entity Formats
- order - Hierarchical format only (ORR)

- item - One Line format only (ITM)
- File group number format: _API_{username}_yyyymmddHHMMSSffffff
- Where ffffff is micro-seconds
- Example: _API_mrafalko_20161209112224184531

Supported Entities

Name	Entity	Version
Vendor	vendor	7.0.0
Appointment	appointment	7.0.0
Item	item	7.0.0
Item Barcode	item_barcode	7.0.0
Site	site	7.0.0
Store	store	7.0.0
Purchase Order	purchase_order	7.0.0
IB Shipment	ib_shipment	7.0.0
Order	order	7.0.0
Work Order	work_order	7.0.0
Planned OB Load	planned_ob_load	7.0.2
Item Prepack	item_prepack	7.0.3
Consolidation Location	consolidation_location_config	8.0.0
IB Shipment Serial Number	ib_shipment_serial_nbr	8.0.0
Item Facility	item_facility	8.0.0
Order Instructions	order_instructions	8.0.0
Literals	literals	18C
Outbound Load	Outbound_load	18C
Movement Request	movement_request	22B

Assign OBLPN to Load

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/assign_oblpn_to_load/"

Initial WMS Version 7.0.0

Overview

API to assign OBLPN(s) to a new or existing Load.

Assumptions

1. If load does not exist, a new one will be created
2. If the load exists in status Shipped or Cancelled, the load will be reused
3. If the load exists in greater than Created status, but less than Shipped status, an error will be thrown
4. Supports bulk mode if more than one OBLPN given in "oblpn_nbr" parameter, separated by character defined in "delimiter" parameter
5. OBLPN must be less than Loaded status
6. If you have the "reassign_load_flg" parameter set to True, then this will not apply as the LPN will be unloaded first.

Request Arguments

Parameter	Description	Initial Version	Required	Data Type	Default
oblpn_nbr	Delimited list of OBLPN numbers	-	X	string	
company_code	WMS company code	-		string	User's default context
facility_code	WMS facility code	-		string	User's default context
trailer_nbr	Trailer updated on the load	-		string	""
carrier_code	Carrier updated on the load	-		string	""
reassign_load_flg	Reassign the OBLPN to new load, if already assigned to different load	-		boolean	True
require_specific_ oblpn_status	Validate OBLPN has matching status (Default is Packed)	-		integer	80
delimiter	Character used to separate oblpn_nbr list	-		string	|

Parameter	Description	Initial Version	Required	Data Type	Default
load_nbr	Load OBLPN(s) will be assigned to	-	X	string	

Note: Since version 9.0.0, new company parameter max_allowed_wt_vol_dim_decimal_scale controls the decimal precision for the following weight fields: oblpn_weight.

Request Options Parameters

Name	Required	Type	Default	Description
assign_load_to_partial_order		Boolean	False	

Request Body Example

```
{
 "parameters":

{ "facility_id" :648, "company_id" :369, "order_nbr" :"ORDER16" }
,
 "options" :

{ "load_nbr" : "OBL0000003019", "assign_load_to_partial_order": true }
}
```

Create LPN

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/create_lpn/"

Initial WMS Version: 7.0.1

Overview

API to create a single SKU IBLPN and associated inventory.

Can also be used to cross-dock the IBLPN to an OBLPN for a given destination facility.

Assumptions

1. The inventory is created out of thin air - it is not taken from some location in thewarehouse
2. The container created is an IBLPN
3. If cross-dock mode then final container will be an OBLPN of same number
4. Appropriate inventory history records arewritten
5. Inventory attributes are not currently withinscope

Parameter	Description	Initial Supported Version	Required	Data Type	Default
lpn_nbr		-	X	string	
qty		-	X	integer	
company_code	WMS company code	-		string	User's default context

Parameter	Description	Initial Supported Version	Required	Data Type	Default
facility_code	WMS facility code	-		string	User's default context
item_barcode		-	C	string	""
item_alternate_code		-	C	string	""
batch_number		-	C	string	""
expiry_date	YYYYMMDD	-	C	date	""
xdock_lpn_flg	Create container then cross-dock?	-		boolean	True
order_type	Cross-dock only: xdock order type	-	C	string	""
dest_facility_code	Cross-dock only: xdock destination facility	-	C	string	""
drop_locn_barcode	Cross-dock only: Location of final OBLPN	-		string	""
lpn_weight	Created IBLPN's weight	-		decimal	""
lock_code	If provided, container will be given lock once created or cross-docked	-		string	""

Additional Notes

- item_barcode or item_alternate_code must be provided. If both are given, item_barcode is evaluated first.
- When xdock_lpn_flg is True:
  - order_type is required
  - dest_facility_code is required
- drop_locn_barcode is of type DROP or STAGING
- If the item characteristics require batch number, then batch_number is required

- If the item metrics require an expiration date, then expiry_date is required, except if you've provided an existing batch number that already has an expiry date. The expiry of an existing batch is preserved, even if expiry_date is given.
- expiry_date is in the format YYYYMMDD
- expiry_date cannot be in the past
- Since version 9.0.0, the company parameter max_allowed_qty_decimal_scale controls the decimal precision for the following quantity fields: qty
- Since version 9.0.0, the company parameter max_allowed_wt_vol_dim_decimal_scale controls the decimal precision for the following weight fields: lpn_weigh

Update Output Interface

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/update_output_interface/"

Initial WMS Version: 8.0.0

Overview

API for external systems to update an output interface record.

This is typically used for external systems to communicate failure and an error message so that it's visible to user in the UI.

Also allows triggering resend of an existing file.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type
filename	Output filename	X		string
facility_code	Output record's facility		User's default facility	string
company_code	Output record's company		User's default company	string
interface_type_code	Interface type		""	string
cust_intf_code	Custom interface identifier		""	string
status_id	Desired status of output record	C		integer
message	Message of output record	C	""	string
run_output_interface_flg	Trigger file resend flag		False	boolean

Note:

- status_id, message, or both are required
- company_code, facility_code, interface_type_code, and cust_intf_code are used to identify a unique record
- Valid statuses: Ready (10), Processed (90), Failed (101), and Cancelled (99)
- If the status is set to Ready (10) and no message is provided, any existing message is cleared
- run_output_interface_flg is only valid for records in status Ready (10)
- This mimics the same functionality as pressing the "Resend" button in Output Interface UI

Induct LPN

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/induct_lpn/"

Initial WMS Version: 8.0.0

Overview

Gives the ability for automated systems (MHE) to induct an LPN to a Drop location tied to an MHE conveyor system.

This will trigger the MHE Route Config rules to generate an appropriate Route Instruction message in Output Interface.

The logic mimics that of RF Induct LPN transaction.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type
xml_data	Required data in XML format	C		string
flat_data	Required data in delimited format	C		string

Note: Either xml_data or flat_data must be provided.

Data Format

Field Name	Function	Required	Data Type
facility_code	Container's facility		string
company_code	Container's company		string
lpn_nbr	Container to be inducted	X	string
induct_location	Barcode of induction location	X	string

Note:

- Flat (pipe-delimited) data is valid when using flat_data input argument
- Data must follow the order specified above in the format: facility_code|company_code|lpn_nbr|induct_location
- Multiple containers are separated by a new line
- XML data is valid when use xml_data input argument
- SEE EXAMPLE BELOW
- The "Header" section may be omitted - it is not used at this time
- If facility_code and/or company_code are not provided, the requesting user's default context is used
- The user must be eligible for all facility/company combinations
- The LPN can be Inbound or Outbound
- The induct_location must be of type DROP with an MHE Conveyor system configured
- The base induction logic is used (same as RF Induct LPN):
- An MHE Message must be active for "LPN ROUTE" for the MHE system
- Valid container statuses for induct are controlled via Facility Parm "ALLOWED_LPN_STATUSES_TO_MHE_INDUCT"

```
Example XML
<LgfData>
 <Header>
 <DocumentVersion>20B</DocumentVersion>
 <OriginSystem>Host</OriginSystem>
 <ClientEnvCode>wmsdev</ClientEnvCode>
 <ParentCompanyCode>*</ParentCompanyCode>
 <Entity>route_instruction</Entity>
 <TimeStamp>2019-01-25T12:34:56</TimeStamp>
 <MessageId>1234567890</MessageId>
 </Header>
 <ListOfInductedLpns>
 <lpn_induct>
 <facility_code>FAC001</facility_code>
 <company_code>COM001</company_code>
 <lpn_nbr>LPN12345</lpn_nbr>
 <induct_location>LOCN1234</induct_location>
 </lpn_induct>
 <lpn_induct>
 <facility_code>FAC001</facility_code>
 <company_code>COM002</company_code>
 <lpn_nbr>LPN45678</lpn_nbr>
 <induct_location>LOCN5678</induct_location>
 </lpn_induct>
 </ListOfInductedLpns>
</LgfData>
```

Divert Confirm

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/divert_confirm/"

Initial WMS Version: 8.0.0

Overview

Gives the ability for automated systems (MHE) to confirm that an LPN was diverted or located.

This will trigger the update of the LPN's location as well as possibly completing any putaway allocations.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type
xml_data	Required data in XML format	C		string
flat_data	Required data in delimited format	C		string

Note: Either xml_data or flat_data must be provided.

Data Format

Field Name	Function	Required	Data Type
facility_code	Container's facility		string
company_code	Container's company		string
mhe_system_code	MHE system code that did the divert	X	string
lpn_nbr	Container that was diverted/ putaway	X	string
divert_lane	The divert lane pushed down	C	string
dest_locn_brcd	Barcode of specific location	C	String
pallet_nbr	Pallet that was diverted/putaway	-	String

Note:

- Flat (pipe-delimited) data is valid when using flat_data input argument
- Data must follow the order specified above in the format: facility_code|company_code|mhe_system_code|lpn_nbr|divert_lane|dest_locn_brcd
- Multiple containers are separated by a new line
- XML data is valid when use xml_data input argument

  See exmaple below:

  The "Header" section may be omitted - it is not used at this time
- If facility_code and/or company_code are not provided, the requesting user's default context is used
- The user must be eligible for all facility/company combinations
- The LPN can be Inbound or Outbound
- divert_lane or dest_locn_brcd must be provided
- When divert_lane is provided:
- System will locate LPN to Drop location in the given facility with the corresponding divert lane configured
- When dest_locn_brcd is provided:
- Locate the LPN to the specific location provided

  If IBLPN:
- Cannot locate to a Consolidation location
- Cannot locate to a Drop location used for IB Sort
- Must be located to a VAS location if LPN requires VAS
- Must be located to a QC location if LPN requires QC
- If putaway allocation exist
- LPN cannot have a Prevent Putaway lock
- If dest_locn_brcd matches the directed location, putaway will be completed
- If dest_locn_brcd does not match the directed location, putaway allocations will be deallocated
- Putaway allocations must be for a single location
- If OBLPN:
- Cannot locate to Active or Reserve
- Must be in status In-Picking, Picked, In-Packing, or Packed

```
Example XML
<LgfData>
 <Header>
 <DocumentVersion>20B</DocumentVersion>
 <OriginSystem>Host</OriginSystem>
 <ClientEnvCode>wms6head</ClientEnvCode>
 <ParentCompanyCode>*</ParentCompanyCode>
 <Entity>divert_confirmation</Entity>
 <TimeStamp>2019-01-25T12:34:56</TimeStamp>
 <MessageId>1234567890</MessageId>
 </Header>
 <ListOfDivertConfirmations>
 <divert_confirmation>
 <facility_code>FAC001</facility_code>
 <company_code>COM001</company_code>
 <mhe_system_code>CONVYR001</mhe_system_code>
 <lpn_nbr>LPN12345</lpn_nbr>
 <divert_lane>DIVERT001</divert_lane>
 <dest_locn_brcd></dest_locn_brcd>
 </divert_confirmation>
 <divert_confirmation>
 <facility_code>FAC001</facility_code>
 <company_code>COM002</company_code>
 <mhe_system_code>CONVYR002</mhe_system_code>
 <lpn_nbr>LPN12345</lpn_nbr>
 <divert_lane></divert_lane>
 <dest_locn_brcd>LOCN1234</dest_locn_brcd>
 </divert_confirmation>
 </ListOfDivertConfirmations>
</LgfData>
```

Load LPN

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/assign_and_load_oblpn/"

Initial WMS Version: 8.0.2

Method: POST

Overview

API to perform loading of outbound LPNs or Pallet

API can also be used to assign LPN's to load and also perform loading

Assumptions

1. If OBLPN is not assigned to a load then it must be assigned to a load before loading
2. If OBLPN is already assigned to a load and load Nbr passed is different from the already assigned load then system will assign it to a different load and perform loading.
3. OBLPN being loaded must belong to an eligible facility and company for the WMS user who is invoking the API
4. OBLPN Weight cannot be negative.
5. API will not support Load by Order Flow.
6. If LPN is associated with a Pallet and API is invoked for the LPN, we do not auto load all the other LPN's on the pallet.
7. If Weight is passed when a Pallet number is passed then weight is ignored.

Request Arguments

Parameter	Description	Initial Supported Version	Required	Data Type	Default
oblpn_nbr	Required if pallet number is not provided.	-	C	string	
pallet_nbr	Required if outbound LPN number is not provided.	-	C	string	
company_code	WMS facility code	-	X	string	If facility code is not provided then users default facility will be considered
company_code	WMS company code	-	X	string	If company code is not provided then users default facility will be considered
load_nbr	outbound load number against which the lpn or pallet needs to be assigned and loaded.	-	X	string	
trailer_nbr	Used to search load number for assignment based on trailer number passed.	-	X	string	
dock_door_nbr	Used to search load number for assignment based on trailer number passed.	-	X	string	
oblpn_weight	Using the API outbound LPN's weight can also be updated	-	X	decimal	

Important Validations:

- Outbound LPN passed should not be less than Packed status.
- If OBLPN is in loaded or Shipped or delivered status then API responds with an error message.

- If neither Load Nbr, Dock or Trailer Nbr are passed and appropriate load number to assign is not found API responds with error.
- If OBLPN number passed is in Packed Status and is also marked for Audit company parameter, "ALLOW_LOAD_SHIP_WITH_AUDIT_PENDING" if set to “No" then API responds with appropriate error.
- While performing Loading of Pallet, if some of the LPN's are not eligible for loading, then loading of pallet fails, Error response will be for the first LPN which encountered error.
- Checks based on lock code assigned to outbound LPN and also stop ship flag on the order will be considered before performing loading.
- If the OB LPN is associated with Order whose stop ship flag is set to true, API responds with appropriate error.
- If instead of OBLPN Nbr, Pallet Nbr is passed as input pallet number passed should be valid for user’s eligible facility and company.
- Outbound LPN's associated with the pallet should be in packed status. If any of the outbound LPN's fails validations, API responds with appropriate error.
- If neither Load Nbr, Dock and Trailer number is not provided, and oblpn_number or pallet_number is not already assigned to a load, respond with error.

Additional Pointers:

- If outbound LPN is not assigned to the load, API also assigns appropriate load based on the load number passed or load number determined from the trailer passed or load determined from the dock door.
- If outbound LPN is already assigned to a load, it’s not mandatory to send load_number or trailer number or dock door number exclusively, LPN will be loaded against the already assigned load.
- Load number provided in the API will be used to assign and load the LPN/pallet, even if the LPN/Pallet is assigned to a different load.
- Load number determined from the trailer number or dock door is used to assign and load the LPN/pallet, even if the LPN/Pallet is assigned to a different load.
- If Pallet or OBLPN is successfully loaded then return a success message "Pallet/OBLPN <Pallet/OBLPN Nbr> successfully loaded".
- If Pallet or OBLPN is successfully loaded then all the OBLPN's must be set to loaded status
- If all OBLPNs for an order are loaded then order is also updated to loaded status.
- If the OBLPN Weight is sent (i.e non blank) then the weight of the OBLPN must be updated.
- If Load Nbr, Dock and Trailer all three are passed then Load Nbr provided in the API takes the precedence for assigning the oblpn/pallet to the load and perform loading.
- If trailer and dock door number is provided, precedence will be given to the load associated to the trailer for assigning the oblpn/pallet to the load and perform loading.
- If trailer number is only provided, WMS will search for an open load for the trailer number for assigning and loading the outbound LPN's or pallet.
- If dock door is only provided, WMS will search for open load for the dock door for assigning and loading the outbound LPN's or pallet.

Entity Update API

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/entity/entity_name/key/sequence_number/"

Method:PATCH

Initial WMS Version: 8.0.2

Overview

API is used for updating certain attributes of an entity. Sometimes the clients would want to update certain fields of an entity like stop ship flag on the order. Entity Update API provides for an ability to modify certain fields on the requested entity.

- Entity name should be provided as part of the URL. URL should also contain the key against which the specific attributes needs to be updated.
- Sequence number key will be required if the updates are being done for supported detail tables.
- API will respond with a success or error message.
- Not all entities are supported, supported entities are mentioned below.
- If facility code and company code is provided search for the entity key is done for the specific facility code and company code. If facility code and company code is not provided then entity key is searched across for the default facility and company for the API invoked users and also the user's eligible facility and company combination.

Assumptions:

- If the fields provided for update in xml_data is not supported API will respond with an error.

Supported Entities for Entity Update

<table>
<tr><th>Entity</th><th>Usage</th></tr>
<tr><td>order_hdr</td><td>Refer to the following section for additional details and columns exposed. For example, to update order related fields:
https:////<xxx.wms.ocs.oraclecloud.com>//<env_name>/wms/api/entity>/order/order001/

xml_data needs to be specified which encapsulates the attributes to be updated. . Below mentioned are the arguments that needs to be passed for updating order when the entity passed is Order.
<table>
<tr><th>Parameter Name</th><th>Required</th><th>Default Value</th><th>Data Type</th><th>Comments</th></tr>
<tr><td>company_code</td><td>Optional</td><td></td><td>string</td><td>WMS Company Code, if not specified, API invoked users eligible companies will be evaluated</td></tr>
<tr><td>facility_code</td><td>Optional</td><td></td><td>string</td><td>User eligible facility code, if not specified, API invoked users eligible companies will be evaluated.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Entity</th><th>Usage</th></tr>
<tr><td></td><td>
<table>
<tr><th>Parameter Name</th><th>Required</th><th>Default Value</th><th>Data Type</th><th>Comments</th></tr>
<tr><td>xml_data</td><td>X</td><td></td><td></td><td>Fields to be updated with specific values for the corresponding entity.</td></tr>
</table>

Refer the below section for additional details and columns exposed.Example to update purchase order dtl related fields
</td></tr>
<tr><td>purchase_order_dtl</td><td>
https:////<xxx.wms.ocs.oraclecloud.com>//<env_name>/wms/api/entity>/purchase_order_dtl/POTST001/1 (will update columns for particular sequence number passed).

https:////<xxx.wms.ocs.oraclecloud.com>//<env_name>/wms/api/entity>/purchase_order_dtl/POTST001/0 (will update all columns on the purchase order detail).

https:////<xxx.wms.ocs.oraclecloud.com>//<env_name>/wms/api/entity>/purchase_order_dtl/POTST001/ (will update all columns on the purchase order detail).

xml_data needs to be specified which encapsulates the attributes to be updated. Below mentioned section provides details about the arguments.
<table>
<tr><th>Parameter Name</th><th>Required</th><th>Default Value</th><th>Data Type</th><th>Comments</th></tr>
<tr><td>company_code</td><td>Optional</td><td></td><td>string</td><td>WMS Company Code, if not specified, API invoked users eligible companies will be evaluated</td></tr>
<tr><td>facility_code</td><td>Optional</td><td></td><td>string</td><td>User eligible facility code, if not specified, API invoked users eligible companies will be evaluated.</td></tr>
<tr><td>xml_data</td><td>X</td><td></td><td></td><td>Fields to be updated with specific values for the corresponding entity.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Entity</th><th>Usage</th></tr>
<tr><td>active_inventory</td><td>Refer the below section for additional details and columns exposed.

• Example to update active_inventory with item not tracking batch numbers, expiry date or any of the attributes(a-g) and quantity down adjusted by 2.
♂ PATCH wms/api/entity/active_inventory/<location_barcode>/?reason_code=value&item_code=value&adjustment_qty= -2
• Example to update active_inventory with item is tracking batch number only and not tracking expiry date or any of the attributes (a-g) and quantity increased by 2.
○ PATCH wms/api/entity/active_inventory/<location_barcode>/?reason_code=value&item_code=value&adjustment_qty= 2&batch_number=value
○ PATCH wms/api/entity/active_inventory/<location_barcode>/?reason_code=value&item_alternate_code=value&adjustment_qty= 2&invn_attr_a=value (This will try to search for inventory in specified location for item passed in item_alternate_code argument with attribute_a value passed in the API and rest of the other fields with blank.

Below mentioned section provides details about the arguments to be passed when Entity is active_inventory.

<table>
<tr><th>Parameter</th><th>Required</th><th>Data Type</th><th>Default</th><th>Comments</th></tr>
<tr><td>location</td><td>X</td><td>string</td><td></td><td>Location barcode - Passed in URL</td></tr>
<tr><td>reason_code</td><td>X</td><td>string</td><td></td><td>Reason Code provided will be updated on the corresponding inventory history record generated</td></tr>
<tr><td>facility_code</td><td></td><td>string</td><td>User's default facility</td><td>User eligible facility code</td></tr>
<tr><td>company_code</td><td></td><td>string</td><td>User's default company</td><td>User eligible company code</td></tr>
<tr><td>item_code</td><td>C</td><td>string</td><td></td><td>Only one of item_code or item alternate code or item barcode is required</td></tr>
<tr><td>item_alternate_code</td><td>C</td><td>string</td><td></td><td>Only one of item_code or item alternate code or item barcode is required</td></tr>
<tr><td>item_barcode</td><td>C</td><td>string</td><td></td><td>Only one of item_code or item alternate code or</td></tr>
</table>
</td></tr>
</table>

Entity	Usage

Parameter	Required	Data Type	Default	Comments
				item barcode is required
adjustment_qty	C	numeric		Non-zero value. Only one of adjustment qty or actual_qty needs to be provided.
actual_qty	C	numeric		Non-zero value. Only one of adjustment qty or actual_qty needs to be provided.
batch_number		string		New or existing batch tied to the inventory. If item is tracking batch number, batch_number argument needs to be passed.
expiry_date	C	date		Required if a new batch is being created and item is tracking expiration date
invn_attr_a		string		Used to filter target inventory for update.
invn_attr_b		string		Used to filter target inventory for update.
invn_attr_c		string		Used to filter target inventory for update.
invn_attr_d		string		Used to filter target inventory for update.
invn_attr_e		string		Used to filter target inventory for update.

Entity	Usage

Parameter	Required	Data Type	Default	Comments
invn_attr_f		string		Used to filter target inventory for update.
invn_attr_g		string		Used to filter target inventory for update.

If the corresponding item/batch/expiry/inventory attributes do not exist in the specified location system does create a new inventory record for the location, If record found then current quantity can be passed.

Order

The following table describes the fields to be passed in xml_data argument:

Order

Field	Supported Version
stop_ship_flg	8.0.2 (can pass value of true or false)
cust_field_1	8.0.2
cust_field_2	8.0.2
cust_field_3	8.0.2
cust_field_4	8.0.2
cust_field_5	8.0.2
cust_long_text_1	8.0.2
cust_long_text_2	8.0.2
cust_long_text_3	8.0.2
cust_short_text_1	8.0.2
cust_short_text_2	8.0.2
cust_short_text_3	8.0.2
cust_short_text_4	8.0.2
cust_short_text_5	8.0.2
cust_short_text_6	8.0.2

Field	Supported Version
cust_short_text_7	8.0.2
cust_short_text_8	8.0.2
cust_short_text_9	8.0.2
cust_short_text_10	8.0.2
cust_short_text_11	8.0.2
cust_short_text_12	8.0.2

purchase_order_dtl

Below mentioned describes the fields to be passed in xml_data argument:

Purchase Order Detail

Field	Supported Version
cust_field_1	8.0.2
cust_field_2	8.0.2
cust_field_3	8.0.2
cust_field_4	8.0.2
cust_field_5	8.0.2
stop_recv_flg	8.0.2 (can pass value of true or false)

From MHE Distribution Pack

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/from_mhe_distribution_pack/"

Initial WMS Version: 8.0.2

Method: POST

Overview

Rest API to perform packing updates when MHE System is performing distribution and packing of inducted inventory.

New API which provides information related to the outbound LPN's packed by Tilt Tray Sorter or Put to Light System or Distribution Sorter. Once outbound LPN is completely packed MHE system makes an API call to perform packing updates for LPN distributed.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type
xml_data	Required data in XML format	C		string

- API can be invoked by sending xml_data
- API can be invoked with packing information for one for more outbound LPNs.

Parameters

The following section describes the data elements that need to be passed in the xml_data. Information for one or more outbound LPNs to be packed can be sent via xml data.

Field Name	Function	Required	Data Type	Default
facility_code	If facility code is not sent all relevant allocations to be packed will be searched for all eligible facilities for the API invoked user. If facility code is sent allocations to be packed will be searched for the specific facility passed.		string	Users default facility
company_code	If company code is not sent all relevant allocations to be packed will be searched for all eligible companies for the API invoked user defined in WMS. If company code is sent allocations to be packed will be searched for the specific company.		string	Users default company
mhe_system_code	MHE System code which has performed the packing	R	string	
ob_lpn_nbr	outbound LPN number which is created and packed as part of distribution	R	string	
destination_facility_code	WCS system needs to send the destination facility code information associated with the outbound LPN packed	R	string	
pallet_nbr	WCS can send the pallet number if outbound LPN is palletized post packing. Field is not mandatory. If pallet number is specified, routing will not be performed even if induction location is provided.		string	

Field Name	Function	Required	Data Type	Default
current_location	WCS can send the barcode of the location where the packed outbound LPN is currently located.		string	
induction_location	Once outbound LPN is packed, if the outbound LPN is to be routed, corresponding induction location barcode can be provided. If valid induction location barcode is provided (location with mhe system of type conveyor) then WMS will try to determine the appropriate divert. Induction Location shared should be of type "Drop".		string	
distro_control_number		R	string	
wave_number			string	
ib_lpn_nbr	Specifies the inbound LPN number from which the corresponding item is packed in the outbound LPN. This field will be required for consuming the pending distribution allocations.	R	string	
item_alternate_code	Alternate Code of the sku to be distributed. Either send the item alternate code or associated item parts or item_ barcode	C	string	
item_part_a		C	string	
item_part_b		C	string	
item_part_c		C	string	
item_part_d		C	string	
item_part_e		C	string	
item_part_f		C	string	
item_barcode	WCS can send the specific item_ alternate_code or the corresponding item parts or the item_barcode. Item_	C	string	

Field Name	Function	Required	Data Type	Default
	barcode if sent will be matched with the corresponding barcode of the item or from a corresponding alternate item barcode list.			
batch_number	WCS can send the batch number corresponding to the item to be packed.	R	string	
expiry_date	WCS can send the expiry date for the item to be packed Format : YYYYMMDD000000		date	
invn_attr_a	WCS can send the inventory attribute_ a value corresponding to the item to be packed from inbound LPN		string	
invn_attr_b	WCS can send the inventory attribute_ b value corresponding to the item to be packed from inbound LPN		string	
invn_attr_c	WCS can send the inventory attribute_ c value corresponding to the item to be packed from inbound LPN		string	
invn_attr_d	WCS can send the inventory attribute_ d value corresponding to the item to be packed from inbound LPN		string	
invn_attr_e	WCS can send the inventory attribute_ e value corresponding to the item to be packed from inbound LPN		string	
invn_attr_f	WCS can send the inventory attribute_ f value corresponding to the item to be packed from inbound LPN		string	
invn_attr_g	WCS can send the inventory attribute_g value corresponding to the item packed in the outbound LPN		string	
allocation_uom	Depicts the UOM in which the corresponding inventory is allocated. Valid Values to be passed are UNITS, PACKS, CASES		string	
uom_qty	WCS can send the inventory attribute_f value corresponding to the item packed in the outbound LPN		decimal	
packed_qty	WCS can send the inventory attribute_g value corresponding to the item packed in the outbound LPN	R	decimal	

Field Name	Function	Required	Data Type	Default
close_flg	• This flag allows you to close the intermediate IBLPN and create the distribution allocations from the IBLPN, after a successful close. Set to true/false in the from_mhe_ob_lpn_hdr of the xml (for entity: distribution_pack_confirmation) the close needs to be sent with the last pick of the distribution. • Alternatively, you can use the /pick_confirm/close_lpn in case of the race condition.	C	Boolean	

Assumptions

- If the individual records do fail for any business validations system, the respective errors can be seen in the application.
- API does not perform incremental packing updates, once the message is received, outbound LPN shared in the API will be updated to packed status.

Additional Pointers

- Outbound LPN number passed as part of API will be updated to packed status upon successful processing of a record.
- One outbound LPN can be packed from multiple inbound LPN's, in which case the xml will contain information of all inbound LPN's which got distributed into the corresponding outbound LPN.
- API can be used to pass induction location so that packed outbound LPN information sent from WCS can be subjected for route instruction message generation.
- Once API is invoked and appropriate outstanding allocations determined for the inbound LPN passed, corresponding packed qty shall be reduced from the inbound LPN.
- Relevant Order updates and container detail packed inventory history records shall be written.
- Since version 9.0.0, new company parameter max_allowed_qty_decimal_scale controls the decimal precision for the following fields: uom_qty and packed_qty

From MHE Distribution Short

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/from_mhe_distribution_short/"

Initial WMS Version: 8.0.2

Method: POST

Overview

Rest API to perform shorting updates when MHE System is performing distribution and packing of inducted inventory.

This will trigger the update of WMS to perform shorting related updates.

Request Arguments

Argument Name	Function	Required	Default Value	Data Type
xml_data	Required data in XML format	C		string
flat_data	Required data in delimited format	C		string

- Either xml_data or flat_data must be provided.
- API can be invoked with shorting information for one for more inbound LPN/sku combination. Xml or flat file data shared through API can contain multiple inbound lpn/sku combination for performing shorting updates.

Data Format

Field Name	Function	Required	Data Type	Default
facility_code	If facility code is not sent all relevant allocations to short will be searched for all eligible facility for the API invoked user defined in WMS. If facility code is sent allocations to be shorted will be searched for the specific facility passed.		string	Users default facility
company_code	If company code is not sent all relevant allocations to short will be searched for all eligible companies for the API invoked user defined in WMS. If company code is sent allocations to be shorted will be searched for the specific company.		string	Users default company
mhe_system_code	MHE System code where inventory short was observed	R	string	
ib_lpn_nbr	Inbound LPN number against which the shorting updated has to be performed.	R	string	
destination_facility_code	WCS system to share the destination facility code	R	string	

Field Name	Function	Required	Data Type	Default
	information for which the short needs to be performed.			
item_alternate_code	Alternate Code of the sku to be distributed. Either send the item alternate code or associated item parts or item_barcode	C	string	
item_part_a		C	string	
item_part_b		C	string	
item_part_c		C	string	
item_part_d		C	string	
item_part_e		C	string	
item_part_f		C	string	
item_barcode	WCS can send the specific item_ alternate_code or the corresponding item parts or the item_barcode. Item_barcode if sent will be matched with the corresponding barcode of the item or from a corresponding vendor barcode list.	C	string	
short_qty	If sent the short qty cannot be greater than the outstanding allocation for inbound LPN /sku combination. If the short quantity is NOT sent, the system will generate an error and no updates will take place.	R	decimal	

Field Name	Function	Required	Data Type	Default
distro_control_number	WCS system can send the distro control number corresponding to the ib_ lpn/sku combination for which shorting needs to be performed.		string	
reason_code	Reason_code for performing shorting. Should correspond to a valid reason code in the system		string	
batch_number	WCS can send the batch number corresponding to the item to be shorted		string	
expiry_date	WCS can send the expiry date for the item to be shorted Format : YYYYMMDD000000		date	
invn_attr_a	WCS can send the inventory attribute_a value corresponding to the item to be shorted from inbound LPN		string	
invn_attr_b	WCS can send the inventory attribute_b value corresponding to the item to be shorted from inbound LPN		string	
invn_attr_c	WCS can send the inventory attribute_c value corresponding to the item to be shorted from inbound LPN		string	
invn_attr_d	WCS can send the inventory attribute_d value corresponding to the item packed in the outbound LPN		string	
invn_attr_e	WCS can send the inventory attribute_e value corresponding to the item packed in the outbound LPN		string	

Field Name	Function	Required	Data Type	Default
invn_attr_f	WCS can send the inventory attribute_f value corresponding to the item packed in the outbound LPN		string	
invn_attr_g	WCS can send the inventory attribute_g value corresponding to the item packed in the outbound LPN		string	
wave_number			string	

Assumptions

- If multiple records are shared in single API call, response is sent back to the caller once the API request is made and response is not sent back for every error occurrence.
- Deferred shorting updates is not possible through the API, once the relevant shorting record passes validations, appropriate qty will be reduced from the inbound LPN.

Additional Pointers

- User needs to be eligible for facility and company code passed.
- Inbound lpn number passed should be present in the system for performing shorting and should have relevant outstanding distribution allocations.
- mhe_system code shared should be a valid code configured in WMS.
- Destination facility code should be present in the system and open allocations needs to be present for the destination facility code shorted.
- Item information shared by sending item_alternate_code/item parts or item_barcode should correspond to a valid item for the company code shared or users eligible company list.
- Open allocation for Inbound LPN shared must be associated with the mhe_system_code shared and also the item and associated parts like batch number/expiry date and inventory attributes.
- short_qty passed should not be greater than the pending allocations determined for lpn/item and associated parts.
- Once the relevant record passes the validations
- order qty will be reduced depending upon the order type flag.
- Shorting related updates and relevant inventory history record will be written.
- Since version 9.0.0, new company parameter max_allowed_qty_decimal_scale controls the decimal precision for the following fields: short_qty

Update Carrier LPN Label

URL: "xxx.wms.ocs.oraclecloud.com/env_name/wms/api/update_carrier_lpn_label/"

Method: POST

Initial WMS Version: 8.0.2

Overview

API to update the carrier LPN Label image.

Assumptions

- Label is a required argument for Update Carrier LPN Labe API
- Label is a base64.pdf type
- Carrier_webservice label type would support image or pdf.
- User will be able to send info 1 LPN at a time.

Request Arguments

Arguments	Function	Required	Default Value	Data Type
facility_code	Corresponds to a valid facility code		User default	string
company_code	Corresponds to a valid company code		User default	string
oblpn_nbr	LPN number being routed	X		string
label	Carrier LPN Label Image	X		string
carrier_webservice_label_type	Webservice label type			string

Carrier Webservice Label Type Parameter

The “carrier_webservice_label_type” parameter allows you to specify the web service label type that will be uploaded. This field accepts the following values:

- ZPL
- PDF
- IMAGE

For example: If you are uploading the ZPL code for the Carrier LPN Label, then the label type should be set to ZPL.

Note: If the label is sent without sending the label type, the label type defaults to zpl. You should be sure to match the label and the label type, since these values are not validated against each other. The label is a required field (mandatory).

4 Technical Notes

Technical Notes

This section lists necessary technical information and consist following topics:

API Introduction

API Request

API Response

Web Services and REST Overview

Oracle WMS Cloud Request Example (key-value pairs)

How to Get Started with Oracle WMS Cloud Web Services

API Introduction

An Application Programming Interface (API) is a tool used by applications to provide external applications or users to grant access to specific features of the application. Typically, this involves the passing of some argument data to a web URL (also known as an Endpoint) that has access to the API. For example, within Oracle WMS Cloud application there is an API for invoking the input data processing (init stage interface). In order to accomplish this, the API needs to know things like company, facility, the interface name, and other key pieces of information to execute correctly. Our API's allow the application to expose discrete pieces of functionality to other applications or users in a controlled manner without the need to give access to the entire system and without requiring the user interface to be used.

API Request

Each API is given a specific URL hosted as part of the WMS application. The APIs use HTTPS protocol to receive requests and return a response in much the same way that submitting a form on a website works within a browser. When the "form" is submitted, a call to a URL is made over HTTPS, which has the ability to transmit this data within the request. The data can then be extracted from the received request within the WMS application and used to run the API.

Requirements:

- Must be of type POST
- The Content-Type should be "application/x-www-form-urlencoded"
- This allows the data to be sent in key-value pairs
- Any non-ASCII data must be URL encoded to ensure data integrity
- See https://www.w3schools.com/tags/ref_urlencode.asp
- Any URL reserved characters (; / ? : @ = &) in the data must also be properly encoded to ensure data integrity

- The key-value pairs are represented in the request in the format myurl.com?key1=value1&key2=value2...
- If the reserved characters are not encoded in the data itself, they can be misunderstood to have special meaning and cause data corruption when parsing the request.

API Response

Once the API has completed (successfully or not) within the WMS application, an HTTP response is sent back to the requester. WMS APIs will always return a response. This is similar in the way in which a webpage is returned to a requesting user's browser. However, instead of webpage data, all Oracle WMS Cloud APIs are designed to give a standardized response.

Web Services and REST Overview

This section is intended to give a high level overview of web services, how they work, and how customers use them. Web services are a common method by which machines are able to passing data, files, or invoking a process over the internet using the HTTP protocol. The two main flavors of web services are SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). This section will focus primarily on REST (a service based on REST is called a RESTful service) as it's the web service method used by Oracle WMS Cloud.

Objective of Web Services

- The main object of web services is to provide a window to a resource on a server
- A resource can be a document, picture, video, web page, API, or anything that can be represented in a computer system

Why REST Web Services?

- RESTful services are lightweight, maintainable, and scaleable (all important things for a cloud application)

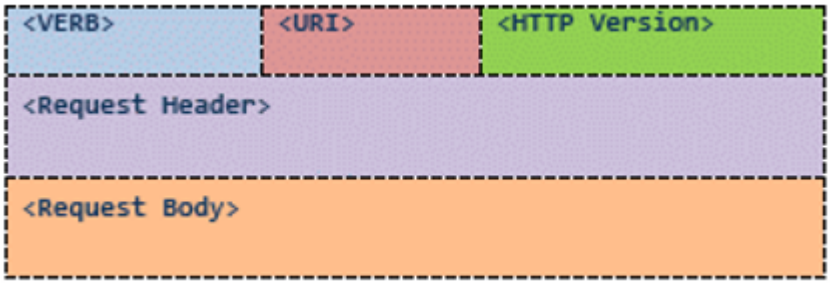
How Do Web Services Utilize HTTP?

- HTTP is the underlying protocol used by web services
- This is the same protocol you just used to request this web page (a resource!) in your browser
- HTTP provides mechanisms to handle the requests and responses to RESTful services
- This includes transferring data and/or files
- The Client/Service/Server Relationship
- A client is the system connecting to and making a request to a service hosted on a server
- The server processes the request, returns a response to the client, and closes the connection

HTTP Messages

- Clients and services talk to each other via messages
- HTTP messages follow a request and response cycle; each request by the client requires a response from the server
- It's possible to not get a response from the server. That typically means there was an issue with the server's execution of the request.

- Requests to a service and responses from a service are both structured messages
- The actual message is just a series of lines of plain text (see Request Example below)



An HTTP request is really nothing more than several lines of text that tell a client about the request it needs to execute.

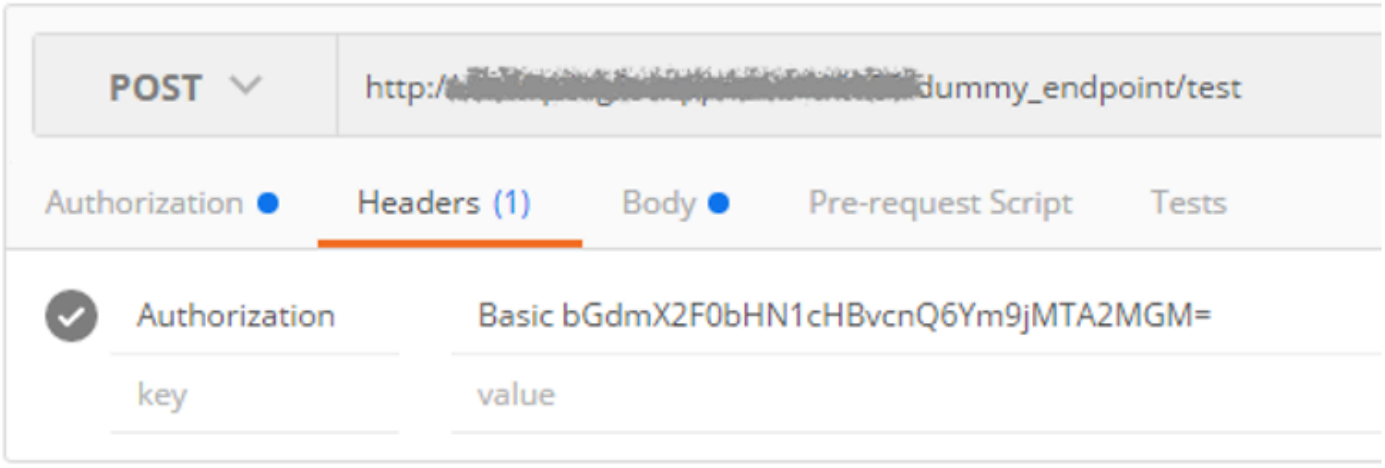
Here we will discuss the components of a request and then walk through an example using the Google Chrome extension, POSTman.

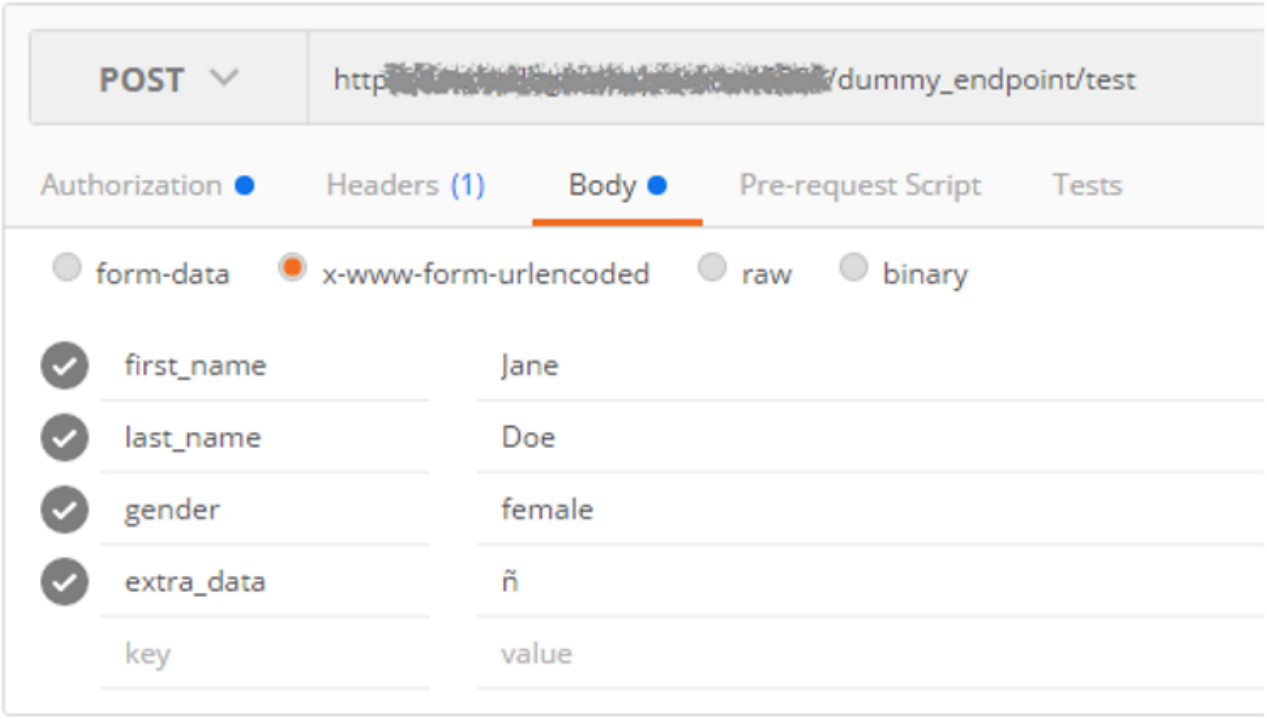
- Verb
- An HTTP method that defines the action of the request
- Examples: GET, POST, PUT, DELETE, ...
- WMS primarily requires clients to POST to our resources
- Some clients and Jitterbit may utilize GET
- URI (Uniform Resource Identifier)
- A URI is a resource on a server that can be accessed by a service
- The most common form of URI is a URL (Uniform Resource Locator)
- A URL specifies both the primary access mechanism and network location
- In simple terms a URL identifies the network and location of the resource being accessed
- Example: http://example.org/wiki/Main_Page
- This URL refers to the resource /wiki/Main_Page that is obtained via HTTP from a network whose domain name is example.org
- HTTP Version
- Current version is "HTTP v1.1"
- Request Header
- Contains request metadata in a collection of key-value pairs
- In general, this is information about the request, the requesting client, authorization, and the format of any data in the request body
- The most important header keys for WMS's purposes are:
- Authorization - An encrypted username/password combination that may be required to access the service
- Content-Type - Defines the format and possibly the encoding (charset) of the data in the request body for POST requests
- The format is known as a MIME Type
- Important POST MIME Types:

- application/x-www-form-urlencoded
- Alphanumeric data is encoded (convert legal non-ASCII characters to a representation using allowed characters) and sent in key-value pairs in the request body
- Any illegal characters, like ñ, are encoded to an ASCII hex representation like "%XX" and then decoded back after transmission
- Example: If you have one field "Name" with a value of "Mary" and another field "Gender" set to "Male", it would be represented as: Name=Mary&Gender=Male
- multipart/form-data
- The data is sent in key-value pairs in the request body in multiple parts
- Typically used for transmitting files (binary data)
- Good for transmitting large amounts of data
- application/xml
- The content of the request body is XML
- Request Body
- The actual content (data) of the message
- Format (and possibly encoding) is determined by the Content-Type header
- Key-value pairs are represented in the format: key1=value1&key2=value2&key3=value3...
- key/value are separated by "="
- pairs are separated by "&"
- However, if for example the Content-Type is set to "application/xml" there would be no key-value pairs, just an XML message

Oracle WMS Cloud Request Example (key-value pairs)

Using the POSTman Google Chrome extension, here's a sample HTTP request:





You can see from the screenshots that we have:

1. A request verb of POST
2. A URI (URL) of http://xxxxxxxxxxxxxx/dummy_endpoint/test
3. An Authorization header (I had put in a username/password and POSTman encrypted it for me)
4. A Content-Type header of "application/x-www-form-urlencoded", which tells us that the data in the request body will be key-value pairs:
    a. Even though you don't see this explicitly in the headers screenshot above, it will be present in the actual request shown below
5. 4-data keys with corresponding values in the request body: first_name, last_name, gender, and extra_data

When we convert this request from the POSTman UI to HTTP:

This is the request information that is actually transmitted!

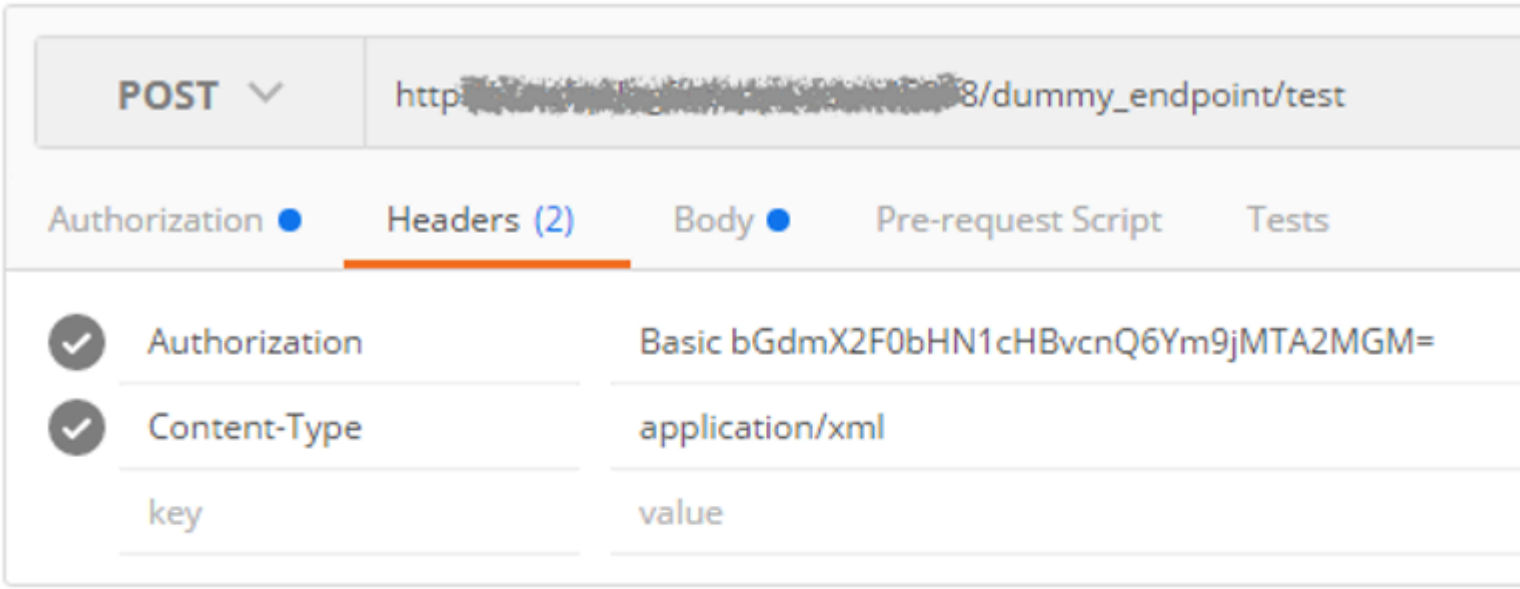
1. It tells HTTP that we want to POST a request to the host xxxxxxxxxxx for resource /dummy_endpoint/test using HTTP version 1.1
2. It also shows that we have the request headers Authorization, Cache-Control, Postman-Token, and Content-Type:
   a. You don't need to worry about Cache-Control or Postman-Token
3. It is shows the request body data as key-value pairs represented in the format discussed
   a. This is expected since the Content-Type is set to "x-www-form-urlencoded"
4. Finally, we can see that it encoded the illegal ñ character to "%C3%B1" for transmission
   a. %C3%B1 is the UTF-8 representation of ñ

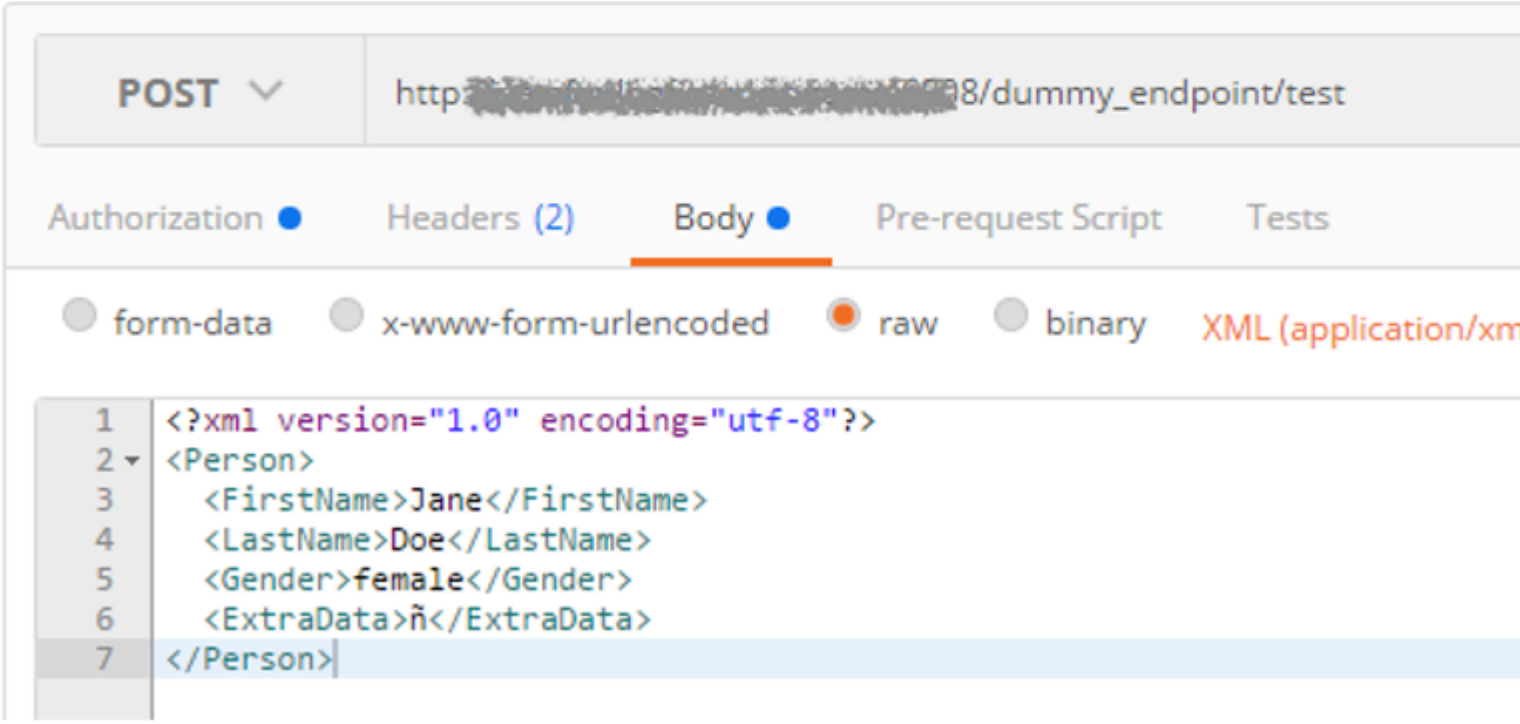
Request Example (XML)

In this example, we will have the same setup as the previous one except that instead of key-value data, we will be sending the XML message:

```
<Person>
 <FirstName>Jane</FirstName>
 <LastName>Doe</LastName>
 <Gender>female</Gender>
 <ExtraData>ñ</ExtraData>
</Person>
```

Using the POSTman Google Chrome extension, the following HTTP request was created:





You can see from the screenshots that we have:

1. A request verb of POST
2. A URI (URL) of http://xxxxxxxxxxxxxxxxxx/dummy_endpoint/test
3. An Authorization header (I had put in a username/password and POSTman encrypted it for me)
4. A Content-Type header of "application/xml", which tells us that the data in the request body will be XML

5. An XML message in the request body

When we convert this request from the POSTman UI to HTTP:

1. It tells HTTP that we want to POST a request to the host xxxxxxxxxxxxxxx for resource /dummy_endpoint/test using HTTP version 1.1
2. It also shows that we have the request headers Authorization, Cache-Control, Postman-Token, and Content-Type:
    a. You don't need to worry about Cache-Control or Postman-Token
3. It is shows the request body XML data:
    a. The XML has been encoded for transmission since characters like "<" and ">" are illegal for HTTP

Notice that the data is the same as before, just represented in an XML format that was made up for this example.

This is done to show that the same data can be passed via many different methods and formats using web services.

What's most important is that the two communicating system agree on these details up front so that each system knows what to expect.

How to Get Started with Oracle WMS Cloud Web Services

This section describes the basic steps necessary to get setup so as to use Oracle WMS Cloud Web Services:

- Login to a Cloud WMS environment with an ADMIN Role user
- Open the Group Configuration screen and create a group with no UI or RF menus
- In the entry field at the top start typing in Group
- Select the group, click the permissions button and in the drilldown screen, select can_run_ws_stage_interface permission and save it
- Open the users screen and copy your user (using the duplicate button which is the button next to the one with the plus sign on the right) and make the following change before saving it:
- Change the Login
- Change the Role from ADMIN to Employee
- Enter a Password and note it down
- Change the employee number
- Change the first and last name
- Use Postman to create a request using the technical notes section and try to post using this new userid and password
- Use asynch=True so that you will get any functional validation errors back
- Open the relevant screen (such as Purchase Order if you're uploading PO's) to check if it loaded
- If not, open the input interface screen and select purchase order to see if there are any errors listed