API Documentation V3.0

1 Changelog

Version	Date	Author	Update
2.1.1	06/01/2016	Jim Potter	- API release
3.0.0	12/28/2016	Jim Potter	- JSON document introduction - Method consolidation & deprecation
3.0.1	02/01/2017	Jim Potter	- Versioning table - Correction to jdDriver xml sample
3.0.2	03/02/2017	Jim Potter	- Updated Customer Field Length
3.0.3	07/11/2017	Jim Potter	- Added deliveryResponse document to allow for resend of deliveryReceipt in case of error/timeout

2 General Overview

At its most basic, JDISPATCH allows delivery drivers to collect an electronic copy of a signature on a touch enabled mobile device. Beyond that, it can also track a driver’s location as well as act as a rudimentary messaging app on a data enabled mobile device. This application, however, will run in a ‘batch mode’ for devices without fill-time data capabilities. Allowing even remotely located customers to take advantage of JDISPATCH’s basic functionality.

2.1 Basic JDISPATCH web service functionality

A common, integrated, JDISPATCH transaction would have a fairly straight-forward flow. An order is placed on the seller's system, and the seller's system processes the order up to the point of being dispatched for delivery. At that point, the seller system generates a dispatch transaction for a driver to send one or more invoices to the JDISPATCH API. Once JDISPATCH receives notice of the delivery, it will notify the driver of a pending delivery and push the order detail to their device. From there, the driver would deliver the product and take a signature of the party receiving the product. That signature is then passed back to the JDISPATCH server where it is saved off and available for future use. Once the signature is saved on the JDISPATCH server, it can optionally be passed back to the sellers system via the API if they wish to merge it on to an electronic copy of the invoice the customer received. After a delivery is made, the confirmation signature is instantly available within the JDISPATCH dashboard for any proof-of-delivery queries that may come in.

In addition, the seller has access to a number of tools to help them analyze their delivery performance. From within the JDISPATCH dashboard, a user sees real-time statistics of average processing and delivery time by driver and customer as well as driver efficiency. Beyond the real time statistics and graphs, the JDISPATCH user has the ability to extract detailed delivery information for analysis after the fact.

Communication between the driver's mobile device and the JDISPATCH server does not have to be on-going. If the mobile device notices a loss of connectivity, it will operate in a 'batch mode' until connectivity is restored, at which point it will send through any deliveries that were processed during the loss of connectivity. Subsequently, any 'batch' transactions will be pushed back to the seller system at the time they are confirmed on the JDISPATCH server.

2.2 JDISPATCH Features

jDispatch Mobile Application	Connected	Batch Mode
Signature capture	Yes	Yes
Part delivery confirmation	Yes	Yes
Background location services	Yes	
System to driver communication	Yes	
Part pickup confirmation	Yes	Yes
‘One Off’ invoice scan and confirm	Yes	
Push notifications	Yes	
Driving directions	Yes	
Route Optimization	Yes	
jDispatch Dashboard	**Connected**	**Batch Mode**
Real-time delivery analysis	Yes	
Real-time corporate delivery metrics	Yes	
Signature recall	Yes	Yes
Delivery analysis extracts	Yes	Yes
Pickup dispatch	Yes	Yes
Driver communication	Yes	
Driver 'on map' location tracking	Yes	
Dashboard configuration	Yes	Yes
Quick phone registration	Yes	Yes

2.3 JDISPATCH Flow Diagram - Integrated

2.4 JDISPATCH Flow Diagram – Non-Integrated

3 Transaction API

3.1 Overview

In order for third-party systems to utilize the integrated features of JDISPATCH, they must utilize the JDISPATCH Transaction API, which allows third-party systems to create dispatch requests, and accept back subsequent delivery confirmations and signatures. The two transaction types utilized are a dispatch transaction and a delivery transaction

3.2 Basic JDISPATCH order transaction architecture

Overall, the JDISPATCH document stack can be broken down in to 2 main segments, delivery dispatch and delivery receipt. These two transactions would be the only **required** integration points for any third party system, as all other functionality of the product can be handled by the JDISPATCH web and mobile applications

1. **Delivery Dispatch:** This is the point where a delivery is dispatched from the third-party system
2. **Delivery Receipt**: This is a transaction that is passed back to a pre-configured URL utilized by the third-party system to take back delivery confirmations once they are processed by the JDISPATCH web application.

3.3 Document Structure

The information passed within a JDISPATCH API document can be sent via XML or JSON. All necessary information within a JDISPATCH API document is contained within an element tag. This is recognized by both a root <jDispatchElement> tag, and an ending </jDispatchElement> tag. Empty elements are allowed within a JDISPATCH API document and will be ignored upon document validation on the JDISPATCH server.

AMS has done their best to select semantic element names to avoid confusion when translating documents. Also, all tag names use Camel Case, indicating that the first letter of each word will be capitalized, and all subsequent letters are lower case. The first letter of the <jDispatchElement> elements will, however, be lower case.

3.4 XML Schema

Each of the JDISPATCH document layouts are described in an XML schema document, or, template. If you are unfamiliar with a schema document, the W3C site provides more information on this topic here: http://www.w3.org/standards/xml/schema. Detailed schemas are listed in Appendix B.

3.5 Document Sections

Each document contains 3 basic elements, a root <jDispatchElement> element, which will be named by the transaction type, either a <jDispatchRequest> or a <jDispatchResponse> <DocHeader> element and a <DocContent> element.

3.6 DocHeader

The header of each document contains identifying information that validates the request and the user sending the request. The required information will not change, regardless of the document type being used. The following attributes **must** exist on each incoming request's header element

Attribute	Required	Description

DocID	Required	A GUID to make each document uniquely identifiable
DocType	Required	Defining the document type that is being sent
Sender	Required	A unique identifier assigned to the sender.
Date	Required	Time stamp of the document, to the millisecond per the ISO 8601 standard. YYYY-MM-DDThh:mm:ss
apikey	Required	API Key provided to you from AMS
env	Required	“TEST” or “PROD”

3.7 DocType Types

Document	Description
deliveryDispatch	Document to create a delivery
deliveryReceipt	Document sent on delivery confirmation
deliveryResponse	Document sent to confirm deliverReceipt
jdCustomer	Document to add a customer to the web
jdDriver	Document To add a driver to the web.
deliveryDispatchResponse	Response from deliveryDispatch document
jdCustomerResponse	Response from jdCustomer document
jdDriverResponse	Response from jdDriver document

3.8 DocContent

The DocContent is much different than the DocHeader element in that the DocContent element will contain all of the information unique to the requested transaction type. A DocContent element may contain many elements of the same type, sending many invoices in one deliveryDispatch document. For example, this allows many invoices to be dispatched at a single time.

XML document examples can be found in Appendix A, and JSON examples can be found in Appendix B, defining element types and requirements.

4 Transaction Overviews

This section will give a general outline of each of the unique transaction types. For detailed XML examples, please refer to Appendix A, and for JSON examples, please refer to Appendix B.

4.1 jDispatchRequest DocType Overviews

deliveryDispatch: The **deliveryDispatch** transaction is the initial dispatch of a delivery to the JDISPATCH web application, and subsequently through to the defined mobile application running on a smart device. Additionally, any cancellations, manual receipts and deletions will be processed via this transaction.

The decision of what type of transaction is being passed is determined by the resquest document's <Status> tag.

A – Add a delivery in to the jDispatch system. Drivers are notified of new deliveries added to their device via a push notification

R – Mark a delivery as Received, this is usually in the case of a delivery being as being picked up or received by the host system

D – Delete a delivery from the jDispatch system. This is usually a scenario where an invoice was inadvertently dispatched, and needs to be 'pulled back' to the host system for dispatch at a later time.

C – Cancel a delivery. This will mark a delivery as cancelled within the jDispatch system. Drivers are notified of cancelled deliveries, and confirmations are done against cancelled deliveries, putting them into a 'Retired' state when confirmed.

Deliveries are sent to the jDispatch system, one delivery batch per driver, and the response of the delivery will be within the <Status> tag of the response, with a response based on the response

status table found in Appendix C, with the detail of any failures being reported in the response's <Message> tag. A successful load of all deliveries will result in a SUCCESS <Status>, with the <Message> tag returning the unique run ID assigned to the delivery batch.

jdCustomer: Any maintenance to customer information (Add/Change/Delete) in the JDISPATCH web application needs to be done via the **jdCustomer** function & transaction. Only basic information is passed for the purpose of enabling the functionality of the JDISPATCH web & mobile applications. This transaction can have multiple customers passed, so bulk actions are possible. The possible statuses for this document are:

A – Add a new customer. Records passed with this status are assumed to be new customers, and the data is added to the system. If, for some reason, the customerID that is passed is already in the system, that customerID will be updated with the detailed information found in the document

D – Delete a customer. The customerID associated with this status will be deleted from the jDispatch system

C – Change a customer. The customer record associated with the customerID passed in this document will have its information updated based on the detail passed within the document.

jdDriver: Any maintenance to driver information (Add/Change/Delete) in the JDISPATCH web application needs to be done via the **jdDriver** function & transaction. Only basic information is passed for the purpose of enabling the functionality of the JDISPATCH web & mobile applications. This transaction can have multiple drivers passed, so bulk actions are possible. The possible statuses for this document are:

A – Add a new driver. Records passed with this status are assumed to be new drivers, and the data is added to the system. If, for some reason, the driverID that is passed is already in the system, that driverID will be updated with the detailed information found in the document

D – Delete a driver. The driverID associated with this status will be deleted from the jDispatch system

C – Change a driver. The driver record associated with the driverID passed in this document will have its information updated based on the detail passed within the document.

4.2 jDispatchResponse DocType Overviews

deliveryReceipt: The **deliveryReceipt** transaction is a transaction sent from the JDISPATCH web application to a pre-configured URL defined by the third-party system (if required). This document passes back the delivery time, signature and recipient for use within the third-party system. The signature image is a JPEG image, encoded with base64 encoding, passed through as an ASCII string and to be decompressed and converted to an image by the host system.

deliveryResponse: The **deliveryResponse** transaction is a transaction sent from the third-party system back to the JDISPATCH api server. This document will pass a status message indicating whether the deliveryReceipt document was accepted and properly processed, and whether to try re-sending. In the case of a failure and re-send request, the JDISPATCH api server will begin attempting to re-send the document. Upon 5 failures, the admin email on the sponsor's account will be notified of the error.

deliveryDispatchResponse: This document will be in the format of the general jDispatchResponse. The basic SUCCESS or FAIL status will be reported by the <Status> tag, a <Code> tag will report a more detailed status with a response based on the response status table found in Appendix C for a **deliveryDispatch** document. Any detailed failure information is contained in the <Message> tag.

jdCustomerResponse: This document will be in the format of the general jDispatchResponse. The basic SUCCESS or FAIL status will be reported by the <Status> tag, a <Code> tag will report a more detailed status with a response based on the response status table found in Appendix C for a **jdCustomer** document. Any detailed failure information is contained in the <Message> tag.

jdDriverResponse: This document will be in the format of the general jDispatchResponse. The basic SUCCESS or FAIL status will be reported by the <Status> tag, a <Code> tag will report a more detailed status with a response based on the response status table found in Appendix C for a **jdDriver** document. Any detailed failure information is contained in the <Message> tag.

4.3 Notable Field Limits

CustomerID – Alpha/Numeric, Limit 15 characters.

DriverID – Alpha/Numeric, Limit 6 Characters.

BranchID = Numeric, Limit 4 digits.

5 Appendix A

5.1 XML Request Transaction Examples

deliveryDispatch

```
<jDispatchRequest>
    <DocHeader>
        <DocID></DocID>
        <DocType>deliveryDispatch</DocType>
        <SenderID></SenderID>
        <apiKey></apiKey>
        <env></env>
        <Date></Date>
    </DocHeader>
    <DocContent>
        <DeliveryCount></DeliveryCount>
        <Delivery>
<BranchID></BranchID>
            <Status></Status>
```

```
        <OrderDate></OrderDate>
        <CustomerID></CustomerID >
        <DriverID></DriverID>
        <Invoice></Invoice>
        <SubTotal></SubTotal>
        <Taxes></Taxes>
        <Freight></Freight>
        <OtherChg></OtherChg>
        <DeliveryType></DeliveryType>
        <Total></Total>
        <Items>
            <LineCode></LineCode>
            <PartNumber></PartNumber>
            <Qty></Qty>
            <Price></Price>
            <Discount></Discount>
            <Extend></Extend>
        </Items>
        <OverrideAddress>
    <Address></Address>
    <City></City >
    <Province></Province>
    <PostalCode></PostalCode>
</OverrideAddress>
```

```
            <SignatureRequired></SignatureRequired>
        </Delivery>
    </DocContent>
</jDispatchRequest>
```

jdCustomer

```
<jDispatchRequest>
<DocHeader>
        <DocID></DocID>
        <DocType>jdCustomer</DocType>
        <SenderID></SenderID>
        <apiKey></apiKey>
        <env></env>
        <Date></Date>
    </DocHeader>
    <DocContent>
<BranchID></BranchID>
        <Customer>
            <Status></Status>
            <CustomerID></CustomerID>
            <CustomerName></CustomerName>
            <Address></Address>
            <City></City>
            <Province></Province>
            <Country></Country>
```

<PhoneNumber></PhoneNumber>

</Customer>

</DocContent>

</jDispatchRequest>

jdDriver

<jDispatchRequest>

<DocHeader>

<DocID></DocID>

<DocType>jdDriver</DocType>

<SenderID></SenderID>

<apiKey></apiKey>

<env></env>

<Date></Date>

</DocHeader>

<DocContent>

<Drivers>

<Status></Status>

```
<BranchID></BranchID>
            <DriverID></DriverID>
            <DriverName></DriverName>
        </Drivers>
    </DocContent>
</jDispatchRequest>
```

5.2 XML Response Transaction Examples

deliveryReceipt

```
<jDispatchResponse>
    <DocHeader>
        <DocID></DocID>
        <DocType>deliveryReceipt</DocType>
        <SenderID></SenderID>
        <apiKey></apiKey>
        <env></env>
        <Date></Date>
    </DocHeader>
    <DocContent>
```

```
        <CustomerID></CustomerID>
        <DriverID></DriverID>
        <DeliveryDate></DeliveryDate>
        <DeliveryTime></DeliveryTime>
        <Recipient></Recipient>
        <SignatureCode></SignatureCode>
        <Delivery>
            <Invoice></Invoice>
        </Delivery>
    </DocContent>
</jDispatchResponse>
```

jdispatchResponse

```
<jDispatchResponse>
    <DocHeader>
        <DocID></DocID>
        <DocType></DocType>
        <SenderID></SenderID>
        <apiKey></apiKey>
        <env></env>
        <Date></Date>
```

```
    </DocHeader>
    <DocContent>
<Status></Status>
<Code></Code>
<Message></Message>
    </DocContent>
</jDispatchResponse>
```

deliveryResponse

```
<jDispatchResponse>
     <DocHeader>
          <DocID></DocID>
          <DocType>deliveryResponse</DocType>
          <SenderID></SenderID>
          <apiKey></apiKey>
          <env></env>
          <Date></Date>
     </DocHeader>
     <DocContent>
<Status></Status>
<Code></Code>
<Message></Message>
```

```
    </DocContent>
</jDispatchResponse>
```

6 Appendix B

6.1 JSON Request Examples

deliveryDispatch

```
{
  "jDispatchRequest": {
    "DocHeader": {
      "DocID": "",
      "DocType": "deliveryDispatch",
      "SenderID": "",
      "apiKey": "",
      "env": "",
      "Date": ""
    },
    "DocContent": {
      "DeliveryCount": "",
      "Delivery": {
```

```
"BranchID": "",
"Status": "",
"OrderDate": "",
"CustomerID": "",
"DriverID": "",
"Invoice": "",
"SubTotal": "",
"Taxes": "",
"Freight": "",
"OtherChg": "",
"DeliveryType": "",
"Total": "",
"Items": {
  "LineCode": "",
  "PartNumber": "",
  "Qty": "",
  "Price": "",
  "Discount": "",
  "Extend": ""
},
"OverrideAddress": {
  "Address": "",
  "City": "",
  "Province": "",
```

```
            "PostalCode": ""
          },
          "SignatureRequired": ""
        }
      }
    }
}
```


jdCustomer


```
{
  "jDispatchRequest": {
    "DocHeader": {
      "DocID": "",
      "DocType": "jdCustomer",
      "SenderID": "",
      "apiKey": "",
      "env": "",
      "Date": ""
    },
    "DocContent": {
      "BranchID": "",
      "Customer": {
```

```
          "Status": "",
          "CustomerID": "",
          "CustomerName": "",
          "Address": "",
          "City": "",
          "Province": "",
          "Country": "",
          "PhoneNumber": ""
        }
      }
    }
}
```


jdDriver


```
{
  "jDispatchRequest": {
    "DocHeader": {
      "DocID": "",
      "DocType": "jdDriver",
      "SenderID": "",
```

```
      "apiKey": "",
      "env": "",
      "Date": ""
    },
    "DocContent": {
      "Driver": {
        "Status": "",
        "BranchID": "",
        "DriverID": "",
        "DriverName": ""
      }
    }
  }
}
```


6.2 XML Response Transaction Examples

deliveryReceipt


```
{
  "jDispatchResponse": {
    "DocHeader": {
      "DocID": "",
```

```
      "DocType": "deliveryReceipt",
      "SenderID": "",
      "apiKey": "",
      "env": "",
      "Date": ""
    },
    "DocContent": {
      "CustomerID": "",
      "DriverID": "",
      "DeliveryDate": "",
      "DeliveryTime": "",
      "Recipient": "",
      "SignatureCode": "",
      "Delivery": {
        "Invoice": ""
      }
    }
  }
}
```


jdispatchResponse


```
{
```

```
  "jDispatchResponse": {
    "DocHeader": {
      "DocID": "",
      "DocType": "",
      "SenderID": "",
      "apiKey": "",
      "env": "",
      "Date": ""
    },
    "DocContent": {
      "Status": "",
      "Code": "",
      "Message": ""
    }
  }
}
```


deliveryReceipt


```
{
  "jDispatchResponse": {
    "DocHeader": {
      "DocID": "",
```

```
        "DocType": "deliveryResponse",
        "SenderID": "",
        "apiKey": "",
        "env": "",
        "Date": ""
      },
      "DocContent": {
      "DocContent": {
        "Status": "",
        "Code": "",
        "Message": ""
      }
    }
}
```


7 Appendix C

7.1 Status Codes

The status codes below are used to report the status of requests sent to this API.

2XX - Success of some kind

4XX - Error occurred in client’s part

5XX - Error occurred in server’s part

Status Code	Description
200	OK
400	Bad request
401	Authentication failure
403	Improper Data Format
404	Invalid Document Type Passed
405	Invalid Document Structure
412	Condition Failed
413	Request Entity Too Large
460	dispatchReceipt failed – Resend requested
465	dispatchReceipt failed – Resend not required
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Revision #4
Created 2018-10-01 15:28:22 UTC by Admin
Updated 2021-08-20 16:45:10 UTC by Jim