Blue Dot

Modern Technology - Simple To Use

Cyan Pickup & Delivery

Data Formats and Integration Outline

Mobile **Fleet Management** Solutions

Table of Contents

About this Document

This document describes the data, message, and node formats used by the Cyan Pickup & Delivery application. It is intended to assist third-party software developers in integrating with the application by creating their own data import and export processes.

Excel Import

Data can be imported from an Excel (.xlsx) spreadsheet using the Data Upload custom page. The file may contain up to four levels of hierarchical data. The names of the four data levels can be configured in the Configuration custom page, but from a technical perspective they are known as trip, stop, order, and line. The sheets in the Excel file are identified by their position; the names given to the sheets are not relevant.
For data to appear in the client, there must be at least stops and orders; trips and lines are optional. If trips are not being used, there must still be an empty sheet at the start of the Excel file containing the Level 1 column headers.
The file will still be imported if it contains fewer than three sheets, but the data won't appear in the client; any sheets after the fourth will be ignored.
The spreadsheet uses an ID column to identify each row. These values should be unique; if data with the same ID as existing data is imported (or if the same spreadsheet is imported again), it will overwrite the existing data.

Required and Supported Columns

The following columns are either required or supported by the data import. Required columns are essential to the application; if they are omitted from the spreadsheet, the import will fail.
Click Refresh in the Data Upload custom page, or check recent Transaction messages in the Web Portal, to see the error message describing the import failure.
Supported columns are used by the application, but the import will still succeed if they are omitted. The application will display blank fields or default values for these columns instead.
The first row in each sheet is assumed to be the column headers. Columns are identified by name; their positions are not important.

Level 1 (e.g. Trip)

Field Name	Field Description	Required (Y/N)
ID	Uniquely identifies each trip. Must be unique within the scope of the tenant the data is being imported into.	Y
AssignedTo	The user name of a user to automatically route this trip to. The user must exist in the tenant the data is being	N

	imported into. This assignment is applied recursively to the trip's stops, orders, and lines, so any AssignedTo values in those rows are ignored. If blank, the data is not initially assigned to any user, and will have to be assigned using the Assignment custom page.	
Description	The trip's title.	N
ScheduledDateTime	The scheduled time of arrival for the trip. This is used to order the trips in the user's Inbox if Sequence is not used.	N
Sequence	If supplied, this number is used to order the trips in the user's Inbox.	N

Level 2 (e.g. Stop)

Field Name	Field Description	Required (Y/N)
ID	Uniquely identifies each stop. Must be unique within the parent trip if ParentID is defined; otherwise, unique to the tenant.	Y
Description	The Stop's title	N
ParentID	Contains the ID value of the parent trip, or leave blank if this is a top-level stop. If specified, the value must correspond to the ID of one of the trips in the first sheet; otherwise, the stop will be treated as a top-level stop.	Y
Address1	Forms part of the stop's address.	N
Address2	Forms part of the stop's address.	N
City	Forms part of the stop's address.	N
State	Forms part of the stop's address.	N
ZipCode	Forms part of the stop's address.	N
Sequence	If supplied, this number is used to order the trips in the user's Inbox.	N
ScheduledDateTime	The scheduled time of arrival for the stop. This is used to order the stops if Sequence is not used.	N

AssignedTo	If ParentID is empty, specifies the user name of a user to automatically route this top-level stop to; otherwise, the stop is assigned to the same user as the parent trip.	N
ContactFirstName	Forms part of the stop's contact details.	N
ContactLastName	Forms part of the stop's contact details.	N
ContactEmail	Forms part of the stop's contact details. The client can display a button allowing the user to send an email to this address.	N
ContactPhone	Forms part of the stop's contact details. The client can display buttons allowing the user to call or text this number.	N
Notes	Free-form text displayed with the stop's details. This is separate from any notes that might be entered by the user.	N

Level 3 (e.g. Order)

Field Name	Field Description	Required (Y/N)
ID	Uniquely identifies each order within the parent stop. If barcode scanning is used to select orders in the client, the barcode must match this ID.	Y
Description	The order's title.	N
ParentID	Contains the ID value of the parent stop.	Y
OrderType	Indicates whether the order is a delivery or pickup. If omitted, the order is assumed to be a delivery. If specified, the value must be either "Delivery" or "Pickup".	Y
QtyScheduled	The quantity expected to be delivered or picked up.	N
UOM	Unit of measure	N
Amount	The pre-tax value of each unit of the order.	N
Tax	The tax added to the Amount for each unit of the order	N
Size/Weight	The dimensions of the order.	

DestinationAddress1	For pickups, forms part of the order's destination address.	N
DestinationAddress2	For pickups, forms part of the order's destination address.	N
DestinationCity	For pickups, forms part of the order's destination address.	N
DestinationState	For pickups, forms part of the order's destination address.	N
DestinationZipCode	For pickups, forms part of the order's destination address.	N

Level 4 (e.g. Line)

Field Name	Field Description	Required (Y/N)
ID	Uniquely identifies each line within the parent order	Y
ParentID	Contains the ID value of the parent order	Y

- *Further columns in Level 4 TBD in future release.*

Custom Fields

Any columns in the spreadsheet that are not "required" or "supported" are considered to be custom fields. They are imported into the node and are displayed in a list by the client.

Identifying Date Values

Excel stores dates and times as numbers – the only difference between a date and a number in Excel is the formatting. Pickup & Delivery will attempt to identify date/time values by looking at the format applied to the cell, but this can only be done reliably if the cell has been formatted using a "standard" Excel date or time format. These are the formats in the Excel Format Cells dialog that correspond to your current regional settings and can be identified by looking for an asterisk (*) prefix.
See http://msdn.microsoft.com/en-us/library/office/documentformat.openxml.spreadsheet.numberingformat(v=office.15).aspx for the list of Excel number formats. The application recognizes formats 14 to 22 as date/times.

As a backup, the application will recognize any cell with a numeric value whose column header contains the text “date” (case-insensitive) as being a date-time value, regardless of the formatting.

Direct Excel Import

The Data Upload custom page works by uploading the Excel file to the server, then posting a Transaction message containing the file details to the Web API. The Pickup & Delivery worker process picks up this message, retrieves the file from the server, and converts it into a Put Data message to be posted back to the Web API.

General Message Structure

All messages in the Cyan Platform share a common JSON structure, with the message-specific content contained in a body attribute that is itself a JSON string. The following example shows a Transaction message created by the Data Upload custom page.


```
{
    "tenantId": "d2a5e576-a952-4abb-8782-74a6f5f006ee",
    "userName": "bluedotuser",
    "messageTypeId": "a50d60c2-ebbd-41bf-b4f3-53220ba6d711",
    "applicationId": "88496d53-8763-4691-b6fb-2aa39136e46e",
    "body": "{\"transactionType\":\"UploadData\",\"otherData\":\"{\\\"fileId\\
\":\\\"074eafe2-62c7-4e76-a2a7-1e4ee3e9380a\\\",\\\"fileName\\\":\\\"Excel Upl
oad 2015-03-05 06.xlsx\\\"}\"}"
}
```


Note that the messageTypeId corresponds to the “Transaction” message type and the applicationId corresponds to Pickup & Delivery. The tenantId must correspond to the identifier used in the X-Cyan-TenantAccessKey header of the request. All of these ID values can be found in the Web Portal, or by calling the Web API. See Accessing Nodes using the Web API for details of the headers that have to accompany the request, and https://api-dev.cyan.us.com/help/api/1/POST-api-messages for more general details.

Transaction Message

For a given uploaded file ID, the Data Upload custom page will post a Transaction message with the following body to the Web API.

To import an Excel file programmatically, without using the custom page, first upload the Excel file to the server by POST-ing it as multipart form data to /api/files. The server responds with JSON data that includes the ID generated for the file.


```
{
    "fileId": "074eafe2-62c7-4e76-a2a7-1e4ee3e9380a",
    "tenantId": "d2a5e576-a952-4abb-8782-74a6f5f006ee",
    "tenantName": "Blue Dot Solutions",
    "name": "Excel Upload 2015-03-05 06.xlsx",
    "size": 13636,
```


```
    "contentType": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    "createdDateUtc": "2015-07-08T20:15:59.773Z",
    "updatedDateUtc": "2015-07-08T20:15:59.773Z"
}
```

Files are opaque to the server, so it's not critical that the content type be exact; it only has to match the encoding used to post the file.

Now submit the Transaction message to the server by POST-ing JSON to /api/messages.

```
{
    "transactionType": "UploadData",
    "otherData": "{\"fileId\":\"0dc690d4-f97e-44fe-b1a2-22651986af16\",\"fileName\":\"PickupDeliveryAssignments_2015-06-10.xlsx\"}"
}
```

The server responds with a copy of the message that includes its ID. You can use this to poll the server for message audits to determine whether the Excel file was read successfully.

```
{
    "messageId": "d29286f9-fad7-4369-ae94-fa4889cafeb4",
    ...
    "messageAudits": [
        {
            "messageAuditId": "a2c5a2fd-bda8-4866-b560-74c391792527",
            ...
            "messageAuditTypeName": "Message Created",
            "text": "Message created in data store.",
        },
        {
            "messageAuditId": "f4a4ea0a-8a7f-4b7b-9726-af7f00a33519",
            ...
            "messageAuditTypeName": "Dispatched by Router",
            "text": "Message sent to service bus queue: WorkerDelivery",
        }
    ],
    "createdDateUtc": "2015-07-08T20:16:03.877Z",
    "updatedDateUtc": "2015-07-08T20:16:06.167Z"
}
```

Creating Required Message Routes

The Cyan Platform uses message routes to direct messages to the correct worker process. To create the required message routes for Pickup & Delivery, access the Data Upload custom page at least once for each tenant. The routes will be automatically created and a message will appear at the top of the page indicating whether they were created successfully or not. If no message appears, the routes already exist.

Direct Node Import

The Pickup & Delivery worker process takes the Transaction messages created by the Data Upload custom page, reads the specified Excel file and generates a Put Data message containing nodes that correspond to the Excel data.

Put Data Message

To import nodes without using Excel, a Put Data message can be POST-ed directly to /api/messages. Using the same message structure as shown in General Message Structure, and the message type ID for "Put Data", post a message body similar to the following:

```
{
    "nodePackages": [
        {
            "node": {
                "tenantId": "d2a5e576-a952-4abb-8782-74a6f5f006ee",
                "type": "CpadTrip",
                "id": "3050003",
                "parentType": "CpadRoot",
                "parentId": "0",
                "title": "Trip 2015-03-05 03",
                "props": {
                    "__import": {
                        "fileName": "Excel Upload 2015-03-05 06.xlsx",
                        "rowNumber": 2,
                        "sheetName": "Level 1 (Trip)"
                    },
                    "description": "Rory 2015-03-05 03",
                    "scheduledDateTime": "2014-12-16T00:00:00.0000000",
                    "status": null,
                    "id": "3050003",
                    "_closed": "false",
                    "_childCount": "2"
                },
                "refs": {},
            },
            "nodeIndices": [
                {
                    "propertyKey": "_closed",
                }
            ],
            "nodeRoutes": [
                {
                    "tenantId": "d2a5e576-a952-4abb-8782-74a6f5f006ee",
                    "userId": "7f4dd99d-1516-4a4f-a4ed-030aba910029",
                    "applicationId": "88496d53-8763-4691-b6fb-2aa39136e46e",
                    "name": "Route",
                }
            ]
        },
```

The Put Data message structure consists of a nodePackages array; each package contains a single node object, an optional nodeIndices array, and an optional nodeRoutes array. See https://api-dev.cyan.us.com/help/api/1/GET-api-messageBodies-putData for details.

See Accessing Nodes using the Web API for details of the headers that have to accompany the request, and Node Structures for the contents of the nodes.
Examples of these messages can be found by searching the Web Portal's Messages page. It may help to know that the worker process uses a user name equal to the tenant's access key, e.g., "blue-dot-solutions".

Generating Unique IDs

The Cyan Platform requires that all node IDs be unique to the tenant; to allow stop IDs to be reused across different parent trips, a unique, idempotent, node ID is generated by concatenating the values of the **ID** and **ParentID** Excel columns with a dash. For example, if a trip's ID was "T001", the stops in that trip might have node id values of "T001-S001" and "T001-S002". The original, unmodified, ID is held in the props.id field for use in the UI.
This process is repeated recursively so that a line might have a node id of "T001-S002-O003-L004".

Required Node Indexes

Node indexes are used by Cyan to search the contents of a node's props object, and the Pickup & Delivery custom pages rely on several for sorting and filtering. When posting nodes in a Put Data message, node indexes must be included for the following properties:

Trip

_closed Identifies data that has been completed. The initial value of the corresponding property must be "false".

Stop

_closed Identifies data that has been completed.

_contactName Concatenates the customer name fields into a single line.

_customerAddress Concatenates the customer address fields into a single line.

Order

_closed Identifies data that has been completed.

Route Nodes to Assigned User

Nodes can be assigned automatically to a user by including a node route in the Put Data message package.

```
"nodeRoutes": [
    {
        "tenantId": "d2a5e576-a952-4abb-8782-74a6f5f006ee",
        "userId": "7f4dd99d-1516-4a4f-a4ed-030aba910029",
```

```
        "applicationId": "88496d53-8763-4691-b6fb-2aa39136e46e",
        "name": "Route",
    }
]
```

The applicationId must identify the Pickup & Delivery application. Routes must be included for the trip or top-level stop, and every descendant node.
Although the message structure allows for multiple routes per node, the Pickup & Delivery application does not support this, and the resulting behaviour would be undefined. Similarly, parent and child nodes must be assigned to the same user.
Data that is not assigned to any user (by omitting node routes from that package) will be available in the Assignment custom page to assign later.

Virtual Root Nodes

Trips and top-level stops must have a parentType of "CpadRoot" and a parentId of "0". No node with that type and id actually exists; the values allow the Assignment custom page to differentiate between top-level data and stops that belong to trips.

Derive Stop Type from Order Types

Stop nodes contain a stopType property that must be derived from its child orders. If all the orders have the same orderType of either "Delivery" or "Pickup", stopType must contain the same value. If the stop contains a mix of order types, stopType must be set to "PickupDelivery".

Node Structures

All Cyan nodes have a common structure with all application-specified properties contained within a props object.
The props content for each node type used in Pickup & Delivery is mostly composed of fields taken from the source Excel spreadsheet. See Excel Import for a list of those columns. The following sections document additional fields that are used by the application, and the nodes used to send results back from the client.

Trip

Trip nodes are identified by a type of "CpadTrip".

__import **Metadata that allows the node to be related back to the spreadsheet it was imported from. This data is not used in the application.**

_childCount **The number of stops in the trip.**

_closed Indicates whether the trip has been completed. Must be either "true" or "false".

Stop

Stop nodes are identified by a type of "CpadStop".

__import Metadata that allows the node to be related back to the spreadsheet it was imported from. This data is not used in the application.

_arrivalDateUtc The date/time entered by the user in the Arrival Confirmation screen.

_childCount The number of orders in the stop. This number may be incremented if orders are added by the user.

_closed Indicates whether the stop has been completed. Must be either "true" or "false".

_contactName Concatenates the customer name fields into a single line.

_customerAddress Concatenates the customer address fields into a single line.

_departureDateUtc The date/time entered by the user in the Departure Confirmation screen.

Order

Order nodes are identified by a type of "CpadOrder".

__import Metadata that allows the node to be related back to the spreadsheet it was imported from. This data is not used in the application.

_childCount The number of lines in the order.

_closed Indicates whether the order has been completed. Must be either "true" or "false".

Line

Line nodes are identified by a type of "CpadLine".

__import Metadata that allows the node to be related back to the spreadsheet it was imported from. This data is not used in the application.

_closed Indicates whether the line has been completed. Must be either "true" or "false".

Stop Completion

Stop completion nodes are identified by a type of "CpadStopCompletion". They are created by the client when the user completes one or more orders, and contain the results for the corresponding stop.

The user may not complete all of a stop's orders at once, so there can be several stop completions that correspond to a single stop. The related stop is identified by the refs object in the stop completion node:

```
"refs": {
    "parent": {
        "type": "CpadStop",
        "id": "12876-91235"
```

```
    }
}
```

childCount The number of order completions that are related to this stop completion.

completedDateUtc The date and time that the user completed this batch of orders.

disposition The text of the disposition selected by the user.

dispositionId The ID value of the disposition selected by the user.

documentCount The number of documents that were added to the stop by the user and submitted along with this stop completion.

noteCount The number of notes that were added to the stop by the user and submitted along with this stop completion.

orderIds A pipe-delimited list of the order IDs that are related to this stop completion. This value is used by the Summary custom page for searching.

photoCount The number of photos that were added to the stop by the user and submitted along with this stop completion.

signature An object describing the customer signature that was captured by the user. Contains the following properties:

- fileName The name of the uploaded signature file. This can be used to get the file ID from the server.

- name The name of the person providing the signature.

- timestamp The date and time the signature was captured.

stopDescription The title of the related stop.

stopId The ID of the related stop.

stopSequence The sequence value from the related stop.

tripDescription The title of the trip that was the parent of the related stop, or null if the stop was a top-level stop.

tripId The ID of the trip that is the parent of the related stop, or null if the stop was a top-level stop.

userName The name of the user who completed this batch of orders.

Order Completion

Order Completion nodes have a type of "CpadOrderCompletion" and identify their parent stop completion node with their parentType and parentId fields. They are created by the client when the user completes an order, and contain the results for that order.

amount The gross value of each unit in the order.

documentCount The number of documents that were added to the order by the user.

id The ID of the related stop.

noteCount The number of notes that were added to the order by the user.

orderType The type of the related order. Either "Delivery" or "Pickup".

photoCount The number of photos that were added to the order by the user.

qtyActual The actual quantity delivered or picked up.

qtyScheduled The expected quantity to be delivered or picked up.

size/Weight The dimensions of each unit in the order.

status The text of the order status selected by the user from the list configured in the Configuration custom page.

statusId The ID value of the order status selected by the user from the list configured in the Configuration custom page.

tax The tax added to the gross value of each unit in the order.

uom Unit of measure.

Stop Status

Stop Status nodes are created by the client every time the user either arrives at or departs from a stop. They have a type of "CpadStopStatus" and identity the stop they belong to with their parentType and parentId fields.

action Indicates the type of status change that occurred. The value is either "arrival" or "departure".

dateTimeUtc The date and time selected by the user.

latitude Forms part of the user's location when they submitted the change.

longitude Forms part of the user's location when they submitted the change.

submittedDateUtc The actual date and time that the user submitted the change.

userName The name of the user who made the change.

Document

Document nodes represent documents captured by the user using their device's camera. They have a type of "CpadDocument". Documents added to stops or orders will relate to the corresponding stop completion or order completion when they are submitted and have a parentType of "CpadStopCompletion" or "CpadOrderCompletion".

Unrelated documents that have been added from the Inbox use a parentType of "CpadDocRoot".

documentType **The text of the document type selected by the user from the list configured in the Configuration custom page.**

documentTypeCode **The ID value of the document type selected by the user from the list configured in the Configuration custom page.**

pageCount **The number of pages in the document.**

pages **Array describing the pages in the document. Each element is an object containing the following properties:**

- bitsPerPixel **The color depth of the page.**

- fileName **The name of the page image file uploaded to the server. This can be used to find the file ID on the server.**

- fileSize **The size of the page image file in bytes.**

- formatThe format of the page image file.

- height The height of the page.

- width The width of the page.

userName **The name of the user who captured the document.**

Note

Note nodes have a type of "CpadNote" and contain free-form text added by the user. A note can be added to stops or orders, and the node's parentType and parentId will identify the corresponding "CpadStopCompletion" or "CpadOrderCompletion".

text **The text entered by the user.**

userName **The name of the user who created the note.**

Photo

Photo nodes have a type of "CpadPhoto" and represent a photo taken and uploaded by the user. A photo can be added to stops or orders, and the node's parentType and parentId will identify the corresponding "CpadStopCompletion" or "CpadOrderCompletion".

fileName **The name of the image file uploaded to the server. This can be used to get the file ID from the server.**

fileSize **The size of the file in bytes.**

Accessing Nodes using the Web API

To export Pickup & Delivery data, use the Cyan Web API to locate nodes and download related file. The Web API is an HTTP-based web service that is accessed using simple HTTP requests. See https://api-dev.cyan.us.com/help/1 for full details.
Every request to the Web API must include the following headers.

X-Api-Version **Set to "1".**

X-Cyan-TenantAccess **The lookup key for the desired tenant.**

Authentication **The Web API uses HTTP Basic authentication. Set this header to the word "Basic" followed by the user name and password, separated by a colon (:), and encoded in base64.**

See https://api-dev.cyan.us.com/help/1 for a list of the URLs and query strings that can be used to access nodes.