Sales Order API Documentation

# Introduction

This document is to be used by 3rd party applications for adding sales orders to Intact iQ. The API is a RESTful API and contains the standard PUT, POST, GET, DELETE commands. This document concentrates on explaining the required endpoints when retrieving or posting a sales order. This document will cover:

* Customers
* Cash Customers
* Customer Delivery Contacts
* Addresses
* Sales Promotions
* Sales Orders

An explanation of each API endpoint will include the following:

1. A description of what it is used for
2. The supporting HTTP headers
3. An Example Request
4. An Example Response
5. An Object Overview Diagram
6. A breakdown of the objects used in the request & response.

# Connecting to the API

To connect to the API your IP address must first be whitelisted on the Behrens firewall. Once this has been approved you can connect to the API using the base URI - <https://api.behrens.co.uk:1026>

To add or retrieve data from Intact iQ you must be issued with an active API Key for security purposes, this will allow you to add & retrieve sensitive data. The API key is passed in every API call made.

More information regarding searching, adding, retrieving records can be found on the API landing page. This includes information around pagination, search attributes and parameters and information relating to all individual endpoints.

# Customers

Each Sales Order must contain a customer, the customer might be a credit account or a cash account. If it is a cash customer, the order must also contain a cash customer (See Cash Customer)

The customer who made the order should be used when posting orders. For example, if coming from a website the customers code is expected to be known by the website when the order is placed.

Due to Behrens new customer set up process new customers will not be able to be created or updated from using the API at the current time.

### Example Request

A specific customer can be found by calling the endpoint:

GET

/Customers?Code[eq]=52235

### Example Response

{

    "TotalCount": 1,

    "Data": [

        {

            "ID": 8594239372443,

            "UpdatedOn": "2020-05-05T16:13:51.94Z",

            "CreatedOn": "2018-01-22T09:16:47.387Z",

            "Code": "52235",

            "Name": "Website Orders - La Beeby",

            "CreditStatus": {

                "ID": 8619999391678,

                "Code": "2 OVERDUE",

                "Description": "Overdue"

            },

            "D\_Division": {

                "ID": 39232911341392714,

                "Code": "LBY",

                "Description": "La Beeby"

            },

           "Type": {

                "ID": 8645769181880,

                "Code": "CASH",

                "Description": "Cash Debtors"

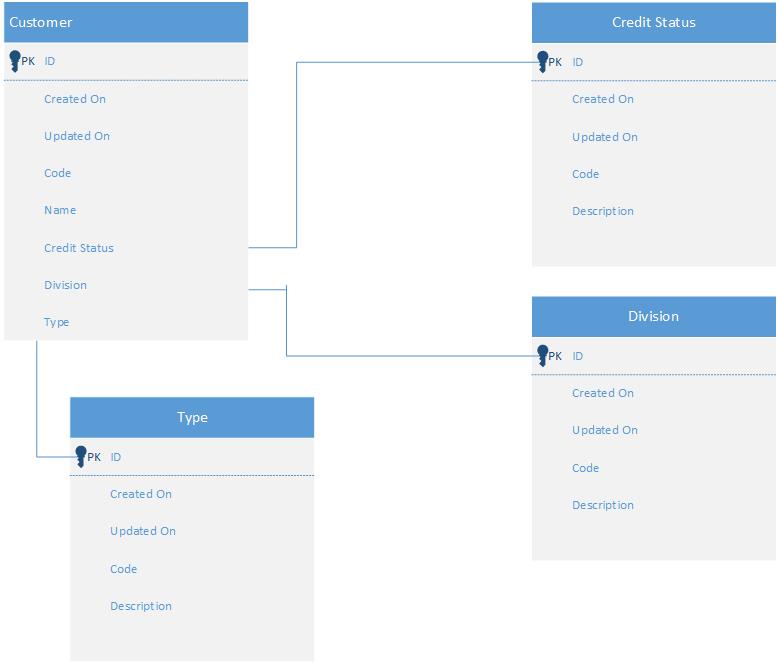
            }

        }

    ]

}

### Object Overview Diagram



# Cash Customers

If the customer is of type “CASH” then the order must also contain a Cash Customer. Cash Customers are a set of customers who don’t require credit. They may either pay for their order upfront, on before the goods are despatched. Cash Customers are linked through a single customer so for example an e-Commerce website will have one iQ Customer but many Cash Customers.

Since the majority of cash customers will have an email address as their log in on a website, the email address field has been selected as the Primary Key. This will need to be used to search for existing cash customers before creating a new one.

### Example Get Request

GET

/CashCustomers?EmailAddress[eq]=rcurran@behrens.co.uk

### Example Get Response

{

    "TotalCount": 2,

    "Data": [

        {

            "ID": 10522690524208,

            "UpdatedOn": "2020-04-28T17:26:15.693Z",

            "CreatedOn": "2018-08-14T10:34:56.053Z",

            "FullName": "Curran,Ryan",

            "FirstName": "Ryan",

            "LastName": "Curran",

            "Salutation": {

                "Value": 1,

                "Name": "Mr"

            },

            "EmailAddress": "rcurran@behrens.co.uk",

            "Phone": "",

            "Mobile": "07507994202",

            "Branch": {

                "ID": 4337916983871,

                "Code": "HQ",

                "Description": "Behrens Group"

            },

            "Address": {

                "ID": 4333642650671

            },

            "D\_Division": {

                "ID": 39232911349295554,

                "Code": "D&D",

                "Description": "Dip & Doze"

            }

        }

    ]

}

If a record exists the response will return a Cash Customer ID. Due to GDPR Legislation the most up ro date personal information of the cash customer needs to be kept within the system. Therefore a Put request is required to update the cash customer

### Example Put Request

PUT

/CashCustomers/10522690524208

{

"FirstName": "Ryan",

"LastName": "Curran",

"EmailAddress": "rcurran@behrens.co.uk",

"D\_Division": "39232911349295554",

"Mobile": "07507994202",

"Branch": "4337916983871",

"Address": "4333642650671"

}

### Example Post Response

{

    "status": "Processed",

    "message": "Ok",

    "queueEntryId": 523170026979923,

    "items": [

        {

            "id": 10522690524208,

            "typeName": "Cash Customer",

            "info": "Curran,Ryan"

        }

    ]

}

If the cash customer doesn’t exist a new cash customer needs to be created.

### Example Post Request

POST

/CashCustomers/

{

"FirstName": "Ryan",

"LastName": "Curran",

"EmailAddress": "rcurran@test.com",

"D\_Division": "39232911374255618",

"Phone": "01618721444",

"Branch": "4337916983871",

"Address": "4333644911379"

}

### Example Post Response

{

"status": "Processed",

"message": "Ok",

"queueEntryId": 523170026900191,

"items": [

{

"id": 10522730449632,

"typeName": "Cash Customer",

"info": "Curran,Ryan"

}

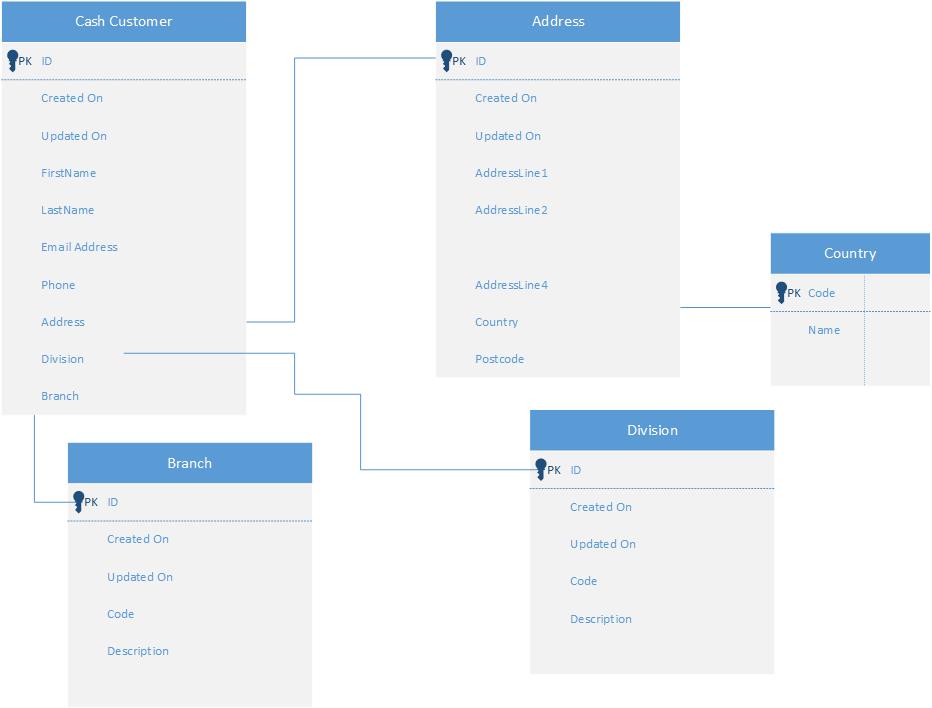
]

}

### Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | bigint | The ID of the object | 10522690524208 |
| FullName | LastName,FirstName | Calculated Field | Curran,Ryan |
| FirstName | String | Customer First Name | Ryan |
| LastName | String | Customer Last Name | Curran |
| Salutation |  | Not Used |  |
| EmailAddress | String (Required) | PK Email Address | rcurran@behrens.co.uk |
| Phone | String | Cash Customer Phone |  |
| Mobile | String | Cash Customer Mobile | 07507994202 |
| Branch | Branch (Required) | Branch | 4337916983871, HQ |
| Address | Address (Required) | Cash Customer Address | 4333642650671 |
| Division | D\_Division (Required) | Cash Customer Division | 39232911374255618, D&D |

### Object Overview Diagram



# Addresses

The Address ID is usually used to retrieve an address.

### Example Get Request

GET

/Addresses/4333644911379

### Example Get Response

{

    "ID": 4333644911379,

    "UpdatedOn": "2020-04-28T17:22:19.45Z",

    "CreatedOn": "2018-09-11T09:15:54.68Z",

    "AddressLine1": "Centrepoint",

    "AddressLine2": "Marshall Stevens Way",

    "AddressLine3": "Trafford Park",

    "AddressLine4": "Greater Manchester",

    "Country": {

        "ID": 4372276717546,

        "Code": "GB",

        "Name": "United Kingdom"

    },

    "PostCode": "M17 1PP"

}

Due to GDPR legislation it is a requirement to try and maintain as accurate personal data within the system. Since I couldn’t determine a suitable way of checking updated addresses against already existing ones. I decided the safest solution would be to post the new address with every order which links to the new or existing record. All countries exist within the system to ISO 3166-1 standard.

### Example Post Request

POST

/SalesDeliveryNotes2/

{

"AddressLine1": "Centrepoint",

"AddressLine2": "Marshall Stevens Way",

"AddressLine3": "Trafford Park",

"AddressLine4": "Greater Manchester",

"Country": "GB",

"PostCode": "M17 1PP"

}

### Example Post Response

{

"status": "Processed",

"message": "Ok",

"queueEntryId": 523170026900282,

"items": [

{

"id": 4333682576187,

"typeName": "Address",

"info": "Centrepoint, Marshall Stevens Way, Trafford Park, Greater Manchester, United Kingdom, M17 1PP"

}

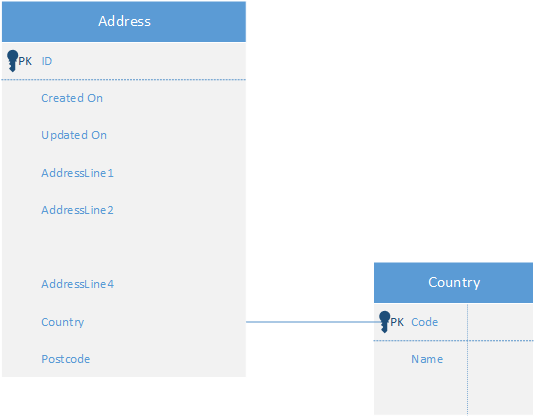
]

}

### Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | bigint | The ID of the object | 4333682576187 |
| AddressLine1 | String (Required) | Address Line 1 | Centrepoint |
| AddressLine2 | String | Address Line 2 | Marshall Stevens Way |
| AddressLine3 | String (Required) | Address Line 3 | Trafford Park |
| AddressLine4 | String | Address Line 4 | Greater Manchester |
| Country | Country (Required) | Country (ISO 3166-1 Standard) | GB |
| Postcode | String (Required) | Postcode | M17 1PP |

### Object Overview Diagram



# Customer Delivery Contacts

Delivery Contacts are linked to the customer (LookupCustomer) and Cash Customer. To determine if the Delivery Contact exists within the system we decided to use the customer email address and their most recent postcode with spaces removed. Due to other systems the FirstName and LastName fields had to be used for this.

### Example Get Request

GET

[/CustomerDeliveryContacts2?lastname[eq]=M171PP&firstname[eq]=rcurran@behrens.co.uk](mailto:/CustomerDeliveryContacts2?lastname[eq]=M171PP&firstname[eq]=rcurran@behrens.co.uk)

### Example Get Response

{

    "TotalCount": 1,

    "Data": [

        {

            "ID": 8680149554231,

            "UpdatedOn": "2019-07-03T15:42:59.993Z",

            "FullName": "rcurran@behrens.co.uk M171PP",

            "FirstName": "rcurran@behrens.co.uk",

            "LastName": "M171PP",

            "CashCustomer": {

                "ID": 10522690524208,

                "FullName": "Curran,Ryan"

            },

            "LookupCustomer": {

                "ID": 8594243213789,

                "Code": "60237",

                "Name": "Website Orders - Dip And Doze"

            },

            "EmailAddress": "rcurran@behrens.co.uk",

            "D\_ContactName": "Ryan Curran",

            "CompanyName": null,

            "Mobile": "07507 994202",

            "Phone": null,

            "Address": {

                "ID": 4333661664992

            }

        }

    ]

}

If the record exists, then a PUT request should only update the relevant details (FirstName, LastName, D\_ContactName, Mobile, Phone and Address). This is due to GDPR legislation we should try and maintain up-to-date information of customers personal information.

### Example Put Request

PUT

/CustomerDeliveryContacts2/8680149554231

{

"FirstName": "rcurran@behrens.co.uk",

"LastName": "M171PP",

"D\_ContactName": "Ryan Curran",

"Address": "4333682576187",

"EmailAddress": "rcurran@behrens.co.uk",

"Phone": "07507994202",

"LookupCustomer": "8594243213789"

}

### Example Put Response

{

    "status": "Processed",

    "message": "Ok",

    "queueEntryId": 523170026980228,

    "items": [

        {

            "id": 8680149554231,

            "typeName": "Customer.Delivery Contact",

            "info": "rcurran@behrens.co.uk M171PP"

        }

    ]

}

If a record doesn’t exist then a Delivery Contact needs to be created, this is done using a Post request.

### Example Post Request

POST

/CustomerDeliveryContacts2/

{

"FirstName": "rcurran@test.com",

"LastName": "M171PP",

"D\_ContactName": "Ryan Curran",

"CashCustomer": "10522730449632,

"Address": "4333682576187",

"EmailAddress": "rcurran@test.com",

"Phone": "01618721444",

"LookupCustomer": "8594243213789"

}

### Example Post Response

{

"status": "Processed",

"message": "Ok",

"queueEntryId": 523170026900469,

"items": [

{

"id": 8680189479926,

"typeName": "Customer.Delivery Contact",

"info": "rcurran@test.com M171PP"

}

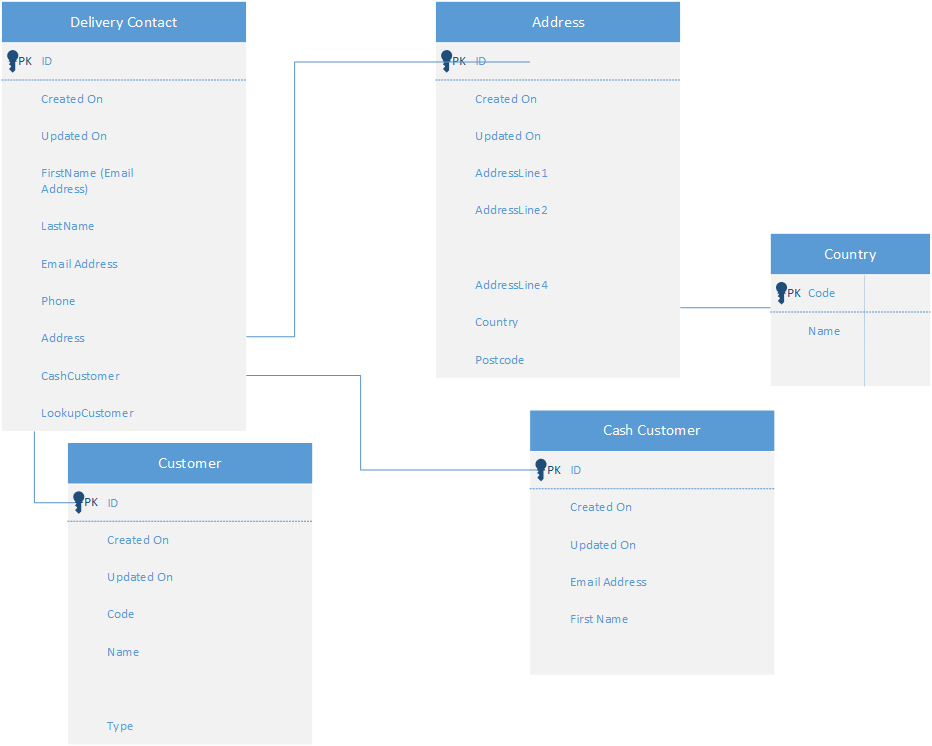
]

}

### Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | bigint | The ID of the Delivery Contact | 8680189479926 |
| FullName | String | Calculated Field | rcurran@behrens.co.uk M171PP |
| FirstName | String (Required) | Customer Email Address | rcurran@behrens.co.uk |
| LastName | String (Required) | Dellivery PostCode no Spaces | M171PP |
| CashCustomer | CashCustomer | Linked Cash Customer (If required) | 10522690524208, rcurran@behrens.co.uk |
| LookupCustomer | Customer (Required) | Customer | 8594243213789, D&D Website |
| EmailAddress | String | Customer Email Address | rcurran@behrens.co.uk |
| D\_ContactName | String | Delivery Contact Name (If Exists) | Ryan Curran |
| CompanyName | String | Delivery Company Name (If Exists) | Dip & Doze |
| Mobile | String | Delivery Contact Mobile | 07507994202 |
| Phone | String | Delivery Contact Phone |  |
| Address | Address (Required) | Address ID | 4333661664992 |

### Object Overview Diagram



# Sales Promotions

If an order has a promotion code it is required that a promotion needs to be added or is already stored within the system. First the promotion code needs to be checked against the existing records in the system.

### Example Get Request

GET

/SalesPromotions?Code[eq]=L\_WELCOME1

### Example Get Response

{

"ID": 9921388238485,

       "UpdatedOn": "2019-12-10T13:40:37.52Z",

       "CreatedOn": "2018-03-20T10:15:53.81Z",

       "Code": "L\_WELCOME1",

       "Description": "Welcome 10% Off LaBeeby"

}

I didn’t think updating promotions was necessary between the systems. However, adding promotions is a necessity. Whether this is managed when the promotion is created within the 3rd party system or when an order is added depends on the systems capabilities.

### Example Post Request

POST

/SalesPromotions/

{

"Code": "RCTEST",

"Description": "TestPromotion"

}

### Example Post Response

{

    "status": "Processed",

    "message": "Ok",

    "queueEntryId": 523170026980261,

    "items": [

        {

            "id": 9921435108262,

            "typeName": "Sales Promotion",

            "info": "RCTEST"

        }

    ]

}

### Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | bigint | The ID of the Promotion | 9921435108262 |
| Code | String (10) (Required) | Promotion Code | RCTEST |
| Description | String | Promotion Description | TestPromotion |

# Sales Orders

Finally, once all the objects have been added the order can be posted. Due to rules within the system an order with the same reference cannot be posted, however, you could perform your own check to see whether orders are in the system.

### Example Get Request

GET

/SalesOrders?AlternateReference[eq]=DOZE\_115373555

### Example Get Response

{

"ID": 9234240331696,

"UpdatedOn": "2020-05-05T14:05:08.19Z",

"CreatedOn": "2020-05-05T14:05:08.19Z",

"Number": "SO0139037",

"AlternateReference": "DOZE\_115378109",

"Customer": {

"ID": 8594243213789,

"Name": "Website Orders - Dip And Doze",

"Code": "60237"

},

"CashCustomer": {

"ID": 10522714522384,

"FullName": "Ryan Curran,rcurran@behrens.co.uk"

},

"Branch": {

"ID": 4337916983871,

"Code": "HQ",

"Description": "Behrens Group"

},

"DespatchBranch": {

"ID": 4337916983871,

"Code": "HQ",

"Description": "Behrens Group"

},

"D\_Division": {

"ID": 39232911349295554,

"Code": "D&D",

"Description": "Dip & Doze"

},

"DeliveryContact": {

"ID": 8680173552422,

"FullName": "rcurran@behrens.co.uk M17 1PP"

},

"Currency": {

"ID": 4346506903774,

"Code": "GBP",

"Description": "Sterling"

},

"Items": [

{

"ID": 9238535298975,

"Product": {

"ID": 21479169499995,

"Description": "600gsm, Organic & Fairtrade Slate Bath Mat",

"Code": "DIP1019"

},

"Quantity": 4.000000,

"SellingUnits": {

"ID": 21539260997245,

"Description": "Each"

},

"NetPrice": 14.166667,

"DiscountPercentageValue": 0.0000,

"TaxRate": {

"ID": 4325032074116,

"Code": "GB01",

"Description": "20% Resale"

},

"GrossPrice": 17.000000,

"GrossPriceDiscountAmount": 0.000000

}

]

}

It is not currently possible to update orders within iQ automatically however the capability is there. When posting an order numerous fields are compulsory for the order to post, some aren’t and are determined by the system when the order is added due to system settings.

### Example Post Request

POST

/SalesOrders/

{

"AlternateReference": "BEH\_TESTRC",

"OrderType": "9247079131646",

"Customer": "8594243213789",

"Branch": "4337916983871",

"DespatchBranch": "4337916983871",

"CashCustomer": "10522730449632",

"DeliveryContact": "8680189479926",

"D\_Division": "39232911374255618",

"Currency": "4346506903774",

"Promotion": "9921435029274",

"DeliveryAgent": "4771713048715",

"DeliveryAgentService": "4776063191099",

"DeliveryTaxRate": "4325032074116",

"DeliveryGrossAmount": "0",

"Items": [

{

"Product": "DOZE1074",

"Quantity": "1.0000",

"SellingUnits": "Each",

"TaxRate": "4325032074116",

"GrossPrice": "15",

"GrossPriceDiscountAmount": "1.5"

},

{

"Product": "DOZE1071",

"Quantity": "1.0000",

"SellingUnits": "Carton",

"TaxRate": "4325032074116",

"GrossPrice": "70",

"GrossPriceDiscountAmount": "7"

}

]

}

### Example Post Response

{

"status": "Processed",

"message": "Ok",

"queueEntryId": 523170026900928,

"items": [

{

"id": 9234240261608,

"typeName": "Sales Order",

"info": "SO0138887"

}

]

}

### Properties

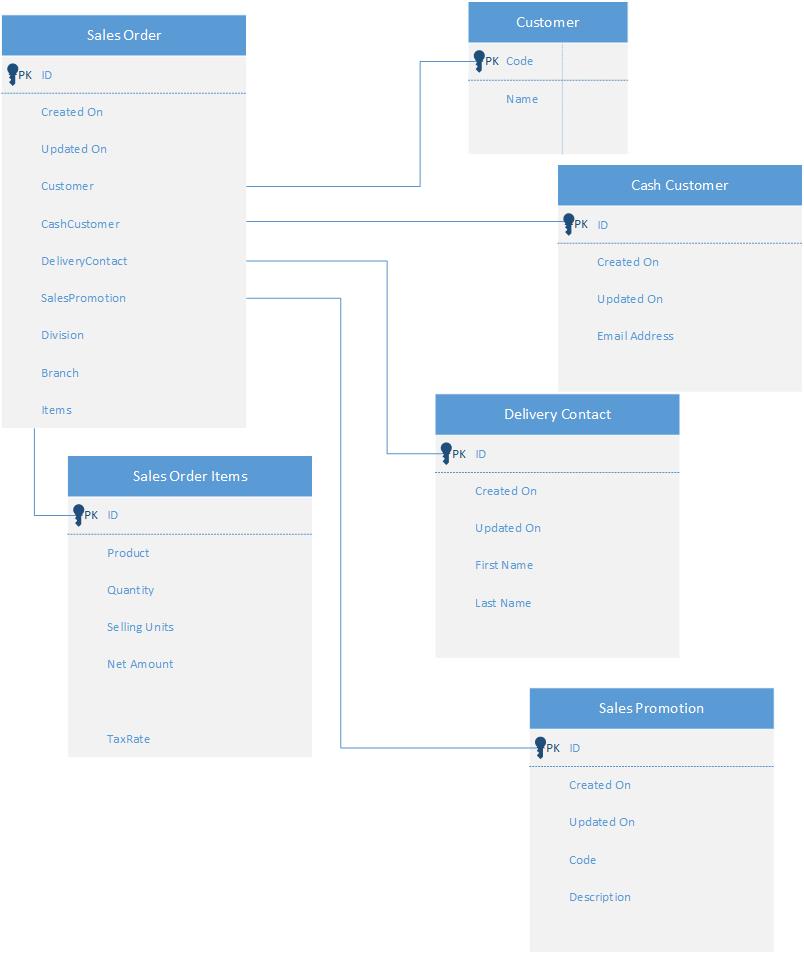
### Sales Order Header

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | Bigint (PK) | The ID of the Sales Order | 9234240331696 |
| Number | String (Unique) | The Intact Order Number | SO0123456 |
| AlternateReference | String (Required) | Customer Reference | BEH\_TESTRC |
| Customer | Customer (Required) | Customer | 8594243213789, Behrens Website |
| CashCustomer | CashCustomer (Required if customer type = cash) | Cash Customer (If Cash Order) | 10522730449632, rcurran@behrens.co.uk |
| Branch | Branch (Required) | Branch | 4337916983871, HQ |
| DespatchBranch | Branch (Required) | Despatch Branch | 4337916983871, HQ |
| D\_Division | D\_Division (Required) | Division | 39232911374255618, D&D |
| DeliveryContact | DeliveryContact (Required) | Delivery Contact | 8680189479926, rcurran |
| Currency | Currency (Required) | Order Currency | 4346506903774, GBP |
| Promotion | SalesPromotion | Promotion |  |
| DeliveryAgent | DeliveryAgent | Delivery Agent (if known) | 4771713048715, DPD |
| DeliveryAgentService | DeliveryAgentService | Delivery Agent Service (if known) | 4776063191099, Next Day |
| DeliveryAgentNetAmount | Numeric | Delivery Net Amount (where net pricing is used) | 10 |
| DeliveryAgentTaxRate | TaxRate | Delivery Tax Rate | 4325032074116, GB01 |
| DeliveryAgentGrossAmount | numeric | Delivery Gross Amount (where gross pricing is used) | 12 |

### Sales Order Items

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Type** | **Description** | **Example** |
| ID | Bigint (PK) | The ID of the Sales Order Line | 9234240331696 |
| Product | Product (Required) | Product ordered | DOZE1074 |
| Quantity | Numeric (Required) | Quantity ordered | 1 |
| SellingUnits | UnitOfMeasure (Required) | Selling Units | Each |
| NetPrice | Numeric (Required if using net pricing) | Net price (where net pricing is used) | 50 |
| DiscountPercentageValue | Numeric | Discount amount (where net pricing is used) | 5 |
| TaxRate | TaxRate | Tax Rate | 4325032074116, GB01 |
| GrossPrice | Numeric (Required where using gross pricing) | Gross price (where gross pricing is used) | 60 |
| GrossPriceDiscountAmount | numeric | Gross Discount Amount (where gross pricing is used) | 6 |

### Object Overview Diagram



# Sales Order API Workflow Diagram

