Circuit Auction Backoffice API Documentation

Base URL

`https://backoffice.ddev.site/api/v1.0/`

Authentication

All API endpoints require authentication using one of the following methods:

Back Office Authentication

Endpoint: `POST /api/v1.0/bo_auth`

Request:

```
{
  "username": "user@example.com",
  "password": "securePassword123"
}
```

Response:

```
{
  "data": {
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
    "user": {
      "id": "1",
      "name": "John Doe",
      "email": "user@example.com",
      "roles": ["administrator"]
    }
  }
}
```

Include the access token in subsequent requests:

`Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc...`

Clients

List Clients (DataTable)

Retrieve a paginated list of clients with advanced filtering and sorting.

Endpoint: `GET /api/v1.0/datatable-clients`

Query Parameters: - `range` - Items per page (default: 25) - `page` - Page number (default: 0) - `sort` - Sort field and direction (e.g., `-id` for descending) - `filter[field_name]` - Filter by field value - `filter[field_name][value]`

- Filter value - `filter[field_name][operator]` - Filter operator (e.g., `IN`, `BETWEEN`, `LIKE`) - `fields` - Comma-separated list of fields to return - `global_search` - Search across multiple fields

Available Filters: - `status` - Client status (new, pending, approved, unapproved, deleted) - `first_name` - First name (regex search) - `last_name_company` - Last name or company name (regex search) - `email` - Email address (regex search) - `phone` - Phone number (regex search) - `country` - Country code (exact match or IN operator) - `city` - City name (regex search) - `state` - State/province (regex search) - `postal_code` - Postal/ZIP code (regex search) - `default_payment_method` - Payment method ID - `source` - Client source/origin ID - `updated` - Last updated date range - `philaworksplace_id` - Customer ID (regex search)

Example Request - Basic List:

```
GET /api/v1.0/datatable-clients?range=25&sort=-updated
```

Example Request - Filter by Status:

```
GET /api/v1.0/datatable-clients?filter[status]=approved&range=50
```

Example Request - Multiple Filters:

```
GET /api/v1.0/datatable-clients?filter[status]=approved&filter[country]=US&sort=-updated&ran
```

Example Request - Search by Name:

```
GET /api/v1.0/datatable-clients?global_search=john&range=25
```

Example Request - Filter by Multiple UUIDs (IN operator):

```
GET /api/v1.0/datatable-clients?filter[uuid][value][0]=abc123&filter[uuid][value][1]=def456&
```

Example Request - Date Range Filter:

```
GET /api/v1.0/datatable-clients?filter[updated][value][from]=2025-01-01&filter[updated][valu
```

Response:


```
{
  "data": [
    {
      "id": "123",
      "uuid": "abc-def-123",
      "first_name": "John",
      "last_name": "Doe",
      "email": "john.doe@example.com",
      "phone": "+1-555-0123",
      "philaworksplace_id": "C12345",
      "status": {
        "id": "approved",
        "label": "Approved"
```

```
      },
      "address": {
        "country": "US",
        "city": "New York",
        "state": "NY",
        "postal_code": "10001",
        "street": "123 Main St"
      },
      "balance": {
        "value": 1250.50,
        "formatted": "$1,250.50"
      },
      "default_payment_method": {
        "id": "3",
        "label": "Credit Card"
      },
      "source": {
        "id": "5",
        "label": "Website"
      },
      "updated": "2025-10-15T14:30:00Z"
    }
  ],
  "count": 342
}
```


Get Single Client

Endpoint: `GET /api/v1.0/datatable-clients/{id}`

Example:

`GET /api/v1.0/datatable-clients/123`

Response:


```
{
  "data": [{
    "id": "123",
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "tax_id": "123-45-6789",
    "categories": [
      {"id": "10", "label": "Modern Art"},
      {"id": "15", "label": "Antiques"}
    ],
    "additional_charges": [],
```

```
    "consignor_commissions": [
      {
        "consignor_commission": "15.00",
        "step_from": "0.00"
      }
    ],
    ...
  }]
}
```

Create Client

Endpoint: `POST /api/v1.0/datatable-clients`

Request:

```
{
  "first_name": "Jane",
  "last_name": "Smith",
  "email": "jane.smith@example.com",
  "phone": "+1-555-0199",
  "status": "new",
  "address": {
    "country": "US",
    "administrative_area": "CA",
    "locality": "Los Angeles",
    "postal_code": "90001",
    "thoroughfare": "456 Oak Ave"
  },
  "source": "5",
  "default_payment_method": "3",
  "categories": ["10", "15"],
  "subscription_type": "email"
}
```

Response:

```
{
  "data": [{
    "id": "456",
    "philaworksplace_id": "C12348",
    ...
  }]
}
```

Update Client

Endpoint: `PATCH /api/v1.0/datatable-clients/{id}`

Request:

```
{
  "status": "approved",
  "phone": "+1-555-0200",
  "categories": ["10", "15", "20"]
}
```

Update Client Status

Endpoint: `PATCH /api/v1.0/datatable-clients/{id}`

Request:

```
{
  "status": "approved"
}
```

Client Connect

Connect two or more client records together.

Endpoint: `POST /api/v1.0/client-connect`

Request:

```
{
  "clients": ["123", "456"],
  "primary_client": "123"
}
```

Client Merge

Merge multiple client records into one.

Endpoint: `POST /api/v1.0/client-merge`

Request:

```
{
  "source_clients": ["456", "789"],
  "target_client": "123",
  "merge_transactions": true,
  "merge_orders": true,
  "merge_items": true
}
```

Response:

```
{
  "data": {
    "merged_client_id": "123",
```

```
    "merged_count": 2,
    "deleted_clients": ["456", "789"]
  }
}
```

Client Address

Get formatted client address.

Endpoint: `GET /api/v1.0/client-address`

Query Parameters: - `client_id` - Client ID

Client Statistics

Endpoint: `GET /api/v1.0/client-stat`

Query Parameters: - `filter[client]` - Client ID - `filter[sale]` - Sale ID (optional)

Response:

```
{
  "data": [{
    "total_orders": 15,
    "total_spent": "12500.00",
    "total_items_won": 42,
    "average_order_value": "833.33",
    "last_purchase_date": "2025-10-01"
  }]
}
```

Connected Entities

Get entities connected to a client.

Endpoint: `GET /api/v1.0/connected-entities`

Query Parameters: - `uuid` - Client UUID

Example:

```
GET /api/v1.0/connected-entities?uuid=abc-def-123
```

Orders

List Orders

Retrieve a paginated list of orders with filtering.

Endpoint: `GET /api/v1.0/orders`

Query Parameters: - `range` - Items per page - `page` - Page number - `sort` - Sort field and direction - `filter[type]` - Order type (bidder, consignor, credit_note, transport_invoice) - `filter[status]` - Order status (new, in_process, finished, locked, canceled) - `filter[sale]` - Sale ID - `filter[owner]` - Client ID - `filter[order_id]` - Order/invoice number - `filter[customer_number]` - Customer number - `filter[total_invoice][value][from]` - Min amount - `filter[total_invoice][value][to]` - Max amount - `filter[payment_received][value][from]` - Min paid amount - `filter[payment_received][value][to]` - Max paid amount - `filter[bidder_ids]` - Bidder number search - `filter[payment_type_letter]` - Payment method - `filter[sale_name]` - Sale name search - `filter[shipping_weight][value][from]` - Min weight (in kg) - `filter[shipping_weight][value][to]` - Max weight (in kg) - `fields` - Specific fields to return

Example Request - Bidder Orders for a Sale:

```
GET /api/v1.0/orders?filter[type]=bidder&filter[sale]=456&filter[status]=new&range=25&sort=-
```

Example Request - Orders with Amount Range:

```
GET /api/v1.0/orders?filter[type]=bidder&filter[total_invoice][value][from]=100&filter[total
```

Example Request - Search by Bidder Number:

```
GET /api/v1.0/orders?filter[bidder_ids]=12345&range=50
```

Response:


```
{
  "data": [
    {
      "id": "789",
      "label": "Order #12345",
      "order_id": "12345",
      "type": "bidder",
      "status": {
        "id": "new",
        "label": "New"
      },
      "sale": {
        "id": "456",
        "label": "Summer Auction 2025"
      },
      "owner": {
        "id": "123",
        "label": "John Doe",
        "email": "john@example.com"
      },
      "grand_total": "1250.00",
      "amount_due": "1250.00",
```

```
      "payment_received": "0.00",
      "customer_number": "C12345",
      "bidder_ids": ["12345", "12346"],
      "item_count": 5,
      "shipping_weight": 2.5,
      "payment_type_letter": "CC",
      "invoice_data": {
        "header_left": {...},
        "header_right": {...},
        "items": {...},
        "totals_for_display": [...]
      },
      "created": "2025-10-01T10:00:00Z",
      "changed": "2025-10-15T14:30:00Z"
    }
  ],
  "count": 42
}
```

Get Single Order

Endpoint: `GET /api/v1.0/orders/{id}`

Example:

`GET /api/v1.0/orders/789`

Response:

```
{
  "data": [{
    "id": "789",
    "order_id": "12345",
    "type": "bidder",
    "status": {
      "id": "new",
      "label": "New"
    },
    "sale": {...},
    "owner": {...},
    "invoice_data": {
      "header_left": {
        "buyer_address": {...},
        "shipping_address": {...}
      },
      "header_right": {
        "invoice_number": "12345",
        "customer_number": "C12345",
```

```
        "bidder_number": "12345",
        "vat_number": "123-45-6789"
      },
      "items": {
        "12345": [
          {
            "nid": "345",
            "lot": "1234",
            "description": "Beautiful antique vase",
            "hammer_price": "250.00",
            "commission": "50.00",
            "tax_id": "6",
            "consignment_number": "A-100"
          }
        ]
      },
      "totals_for_display": [
        [
          {"label": "Subtotal", "value": "$1,000.00"},
          {"label": "Buyer's Premium", "value": "$200.00"}
        ],
        [
          {"label": "Tax", "value": "$50.00"},
          {"label": "Grand Total", "value": "$1,250.00"}
        ]
      ]
    },
    "additional_charges": [
      {
        "term": {
          "id": "5",
          "label": "Shipping Fee"
        },
        "amount": "25.00",
        "tax": "21.00",
        "description": "Standard shipping",
        "date": "2025-10-01"
      }
    ],
    ...
  }]
}
```


Create Order

Endpoint: `POST /api/v1.0/orders`

Request:

```
{
  "type": "bidder",
  "sale": "456",
  "owner": "123",
  "status": "new",
  "items": ["345", "346", "347"],
  "additional_charges": []
}
```

Update Order

Endpoint: `PATCH /api/v1.0/orders/{id}`

Request:

```
{
  "status": "in_process",
  "additional_charges": [
    {
      "term": "5",
      "amount": "25.00",
      "tax": "21.00",
      "description": "Shipping fee"
    }
  ],
  "use_custom_address": true,
  "custom_address": {
    "name_line": "Jane Smith",
    "thoroughfare": "789 Pine St",
    "locality": "Boston",
    "administrative_area": "MA",
    "postal_code": "02101",
    "country": "US"
  }
}
```

Update Order Status

Endpoint: `PATCH /api/v1.0/orders/{id}`

Request:

```
{
  "status": "finished"
}
```

Split Order

Split items from an order into a new or existing order.

Endpoint: `PATCH /api/v1.0/order-split`

Request:

```
{
  "order": "789",
  "items": ["345", "346", "347"]
}
```

Response:

```
{
  "data": {
    "new_order_id": "890",
    "original_order_id": "789",
    "moved_items": ["345", "346", "347"]
  }
}
```

Note: If there's an existing open order for the same client in the same sale, items will be merged into that order instead of creating a new one.

Create Credit Note

Create a credit note for specific items in a finished/locked order.

Endpoint: `POST /api/v1.0/order-create-credit-note/{order_id}`

Request:

```
{
  "items": ["345", "346"]
}
```

Response:

```
{
  "data": {
    "credit_note_order_nid": "891"
  }
}
```

Note: Maximum 100 items per credit note. Order must be in 'finished' or 'locked' status.

Cancel and Reissue Order

Cancel a locked order and create a new one with all items (creates credit note + new order).

Endpoint: `POST /api/v1.0/order-cancel-and-reissue/{order_id}`

Request Body: None required

Response:

```
{
  "data": {
    "new_order_nid": "892",
    "credit_note_nid": "893"
  }
}
```

Note: Only works on locked orders. Used when an invoice needs to be corrected after being exported to accounting system.

Print Order

Generate a printable order/invoice.

Endpoint: `POST /api/v1.0/order-print`

Request:

```
{
  "order_id": "789",
  "signal": "white_bidder_invoice",
  "withDescription": true,
  "with_letterhead": 1
}
```

Signals: - `white_bidder_invoice` - Standard bidder invoice - `accounting` - Accounting format - `via_bulk` - Bulk print format

Orders Attached Text

Get prepend/append text for orders.

Endpoint: `GET /api/v1.0/orders-attached-text`

Query Parameters: - `filter[order]` - Order ID

Reset Additional Charges

Reset order additional charges to match consignment charges.

Endpoint: `PATCH /api/v1.0/reset-statement-charges/{order_id}`

Request Body: None required

Response:

```
{
  "data": {
    "success": true,
    "charges_reset": 3
  }
}
```

Consignments

List Consignments (DataTable)

Retrieve a paginated list of consignments.

Endpoint: `GET /api/v1.0/datatable-consignments`

Query Parameters: - `range` - Items per page - `page` - Page number - `sort` - Sort field and direction - `filter[sale]` - Sale ID - `filter[symbol]` - Consignment symbol - `filter[owner]` - Consignor client ID - `filter[status]` - Consignment status - `fields` - Specific fields to return

Example Request:

```
GET /api/v1.0/datatable-consignments?filter[sale]=456&sort=symbol&range=50
```

Response:


```
{
  "data": [
    {
      "id": "234",
      "label": "Consignment #567",
      "symbol": "A",
      "sub_symbol": "1",
      "sale": {
        "id": "456",
        "label": "Summer Auction 2025"
      },
      "owner": {
        "id": "789",
        "label": "Art Gallery LLC"
      },
      "consignor_commission": "15.00",
      "payment_instructions": "Wire transfer to account...",
      "item_count": 25,
      "total_hammer_price": "15000.00",
      "status": "active"
    }
  ],
```

```
  "count": 15
}
```

Consignment Statements (DataTable)

Get consignment statements.

Endpoint: `GET /api/v1.0/datatable-consignment-statements`

Query Parameters: - `filter[sale]` - Sale ID - `filter[owner]` - Consignor ID - `filter[status]` - Statement status

Consignments Underbids Inquiries

Get inquiries about items that didn't meet reserve/minimum.

Endpoint: `GET /api/v1.0/consignments-underbids-inquiries`

Query Parameters: - `filter[sale]` - Sale ID - `filter[consignment]` - Consignment ID

Control List Consigner

Get control list for consigners.

Endpoint: `GET /api/v1.0/control-list-consigner`

Query Parameters: - `filter[sale]` - Sale ID

Items

List Items (DataTable)

Retrieve a paginated list of items with extensive filtering.

Endpoint: `GET /api/v1.0/datatable-items`

Query Parameters: - `range` - Items per page - `page` - Page number - `sort` - Sort field and direction - `filter[sale]` - Sale ID - `filter[consignment]` - Consignment ID - `filter[category]` - Category ID - `filter[status]` - Item status (available, sold, unsold, passed, withdrawn) - `filter[lot]` - Lot number search - `filter[description]` - Description search (regex) - `filter[estimate_low][value][from]` - Min low estimate - `filter[estimate_low][value][to]` - Max low estimate - `filter[estimate_high][value][from]` - Min high estimate - `filter[estimate_high][value][to]` - Max high estimate - `filter[hammer_price][value][from]` - Min hammer price - `filter[hammer_price][value][to]` - Max hammer price - `filter[consignor_symbol]` - Consignment symbol search - `fields` - Specific fields

Example Request - Items in Sale:

```
GET /api/v1.0/datatable-items?filter[sale]=456&filter[status]=available&range=50&sort=lot
```

Example Request - Sold Items with Price Range:

```
GET /api/v1.0/datatable-items?filter[sale]=456&filter[status]=sold&filter[hammer_price][valu
```

Example Request - Search by Category:

```
GET /api/v1.0/datatable-items?filter[sale]=456&filter[category]=10&range=100
```

Response:


```
{
  "data": [
    {
      "id": "345",
      "label": "Item #1234",
      "lot": "1234",
      "description": "Beautiful antique vase from the Ming Dynasty",
      "sale": {
        "id": "456",
        "label": "Summer Auction 2025"
      },
      "consignment": {
        "id": "234",
        "symbol": "A",
        "sub_symbol": "1"
      },
      "category": {
        "id": "10",
        "label": "Ceramics"
      },
      "estimate_low": "100.00",
      "estimate_high": "200.00",
      "reserve_price": "80.00",
      "hammer_price": "250.00",
      "status": "sold",
      "winner": {
        "id": "123",
        "label": "John Doe",
        "bidder_number": "12345"
      },
      "images": [
        {
          "id": "1001",
          "url": "https://.../files/item-images/vase-1.jpg",
          "thumbnail": "https://.../files/item-images/vase-1-thumb.jpg"
        }
      ],
```

```
      "dimensions": {
        "height": "30cm",
        "width": "15cm",
        "depth": "15cm"
      },
      "created": "2025-09-01T10:00:00Z",
      "updated": "2025-10-15T14:30:00Z"
    }
  ],
  "count": 250
}
```


Get Single Item

Endpoint: `GET /api/v1.0/datatable-items/{id}`

Items List (Simple)

Get a simpler list of items without full datatable features.

Endpoint: `GET /api/v1.0/items-list`

Query Parameters: - `filter[sale]` - Sale ID - `range` - Items per page

Items JSON

Get items in JSON format with specific formatting.

Endpoint: `GET /api/v1.0/items-json`

Query Parameters: - `filter[sale]` - Sale ID - `fields` - Comma-separated field list

Example:

```
GET /api/v1.0/items-json?filter[sale]=456&fields=id,lot,description,estimate_low,estimate_hi
```

Items Public

Get publicly available items (for frontend/catalog).

Endpoint: `GET /api/v1.0/items-public`

Query Parameters: - `filter[sale]` - Sale ID - `range` - Items per page - `sort` - Sort order

Example:

```
GET /api/v1.0/items-public?filter[sale]=456&range=100&sort=lot
```

Items Lots

Get items with lot number information.

Endpoint: `GET /api/v1.0/items-lots`

Query Parameters: - `sale` - Sale ID

Example:

`GET /api/v1.0/items-lots?sale=456`

Response:

```
{
  "data": [
    {
      "id": "345",
      "lot": "1234",
      "position": 1
    },
    {
      "id": "346",
      "lot": "1235",
      "position": 2
    }
  ]
}
```

Items Bulk Operations

Perform bulk operations on multiple items.

Endpoint: `POST /api/v1.0/items-bulk`

Request - Update Status:

```
{
  "operation": "update_status",
  "items": ["345", "346", "347"],
  "values": {
    "status": "withdrawn"
  }
}
```

Request - Update Category:

```
{
  "operation": "update_field",
  "items": ["345", "346"],
  "field": "category",
```

```
  "value": "15"
}
```

Request - Delete Items:

```
{
  "operation": "delete",
  "items": ["345", "346"]
}
```

Items Bulk Edit (Lot Numbers)

Update lot numbers for multiple items.

Endpoint: `PATCH /api/v1.0/items-bulk-edit`

Request:

```
{
  "items": [
    {"id": "345", "lot": "1234"},
    {"id": "346", "lot": "1235"},
    {"id": "347", "lot": "1236"}
  ]
}
```

Clone or Move Items

Clone or move items between sales/consignments.

Endpoint: `POST /api/v1.0/item-clone-or-move`

Request - Move Items:

```
{
  "items": ["345", "346", "347"],
  "operation": "move",
  "target_sale": "457",
  "target_consignment": "235"
}
```

Request - Clone Items:

```
{
  "items": ["345", "346"],
  "operation": "clone",
  "target_sale": "457",
  "target_consignment": "235"
}
```

Response:

```
{
  "data": {
    "success": true,
    "operation": "move",
    "items_processed": 3,
    "new_item_ids": ["445", "446", "447"]
  }
}
```

Item Dimensions

Get/update item dimensions.

Endpoint: `GET /api/v1.0/item-dimensions`

Query Parameters: - `filter[item]` - Item ID

Endpoint: `PATCH /api/v1.0/item-dimensions/{dimension_id}`

Request:

```
{
  "height": "30",
  "width": "15",
  "depth": "15",
  "weight": "2.5",
  "unit": "cm"
}
```

Items Histories

Get item history/audit trail.

Endpoint: `GET /api/v1.0/items-histories`

Query Parameters: - `filter[item]` - Item ID - `sort` - Sort order (default: `-created`)

Response:

```
{
  "data": [
    {
      "id": "5001",
      "item": "345",
      "action": "status_change",
      "old_value": "available",
      "new_value": "sold",
      "user": {
        "id": "10",
        "name": "Clerk Name"
```

```
      },
      "created": "2025-10-15T14:30:00Z",
      "details": "Item sold to bidder 12345 for $250.00"
    }
  ]
}
```


Delete Winning Bid

Remove the winning bid from an item.

Endpoint: `DELETE /api/v1.0/delete-winning-bid/{item_id}`

Request Body: None required

Response:


```
{
  "data": {
    "success": true,
    "item_id": "345",
    "previous_winner": "12345",
    "previous_hammer_price": "250.00"
  }
}
```


Save Single Lot Number

Update a single item's lot number.

Endpoint: `GET /api/v1.0/invoke-function/save_single_lot_number`

Query Parameters: - `params[0]` - Sale ID - `params[1]` - Item position - `params[2]` - Lot number - `params[3]` - Force override (0 or 1)

Example:

```
GET /api/v1.0/invoke-function/save_single_lot_number?params[0]=456&params[1]=10&params[2]=12
```

Sales

List Sales (DataTable)

Retrieve a paginated list of sales/auctions.

Endpoint: `GET /api/v1.0/datatable-sales`

Query Parameters: - `range` - Items per page - `page` - Page number - `sort` - Sort field and direction - `filter[status]` - Sale status (draft, published, closed) - `filter[sale_date][value][from]` - Min sale date

- `filter[sale_date][value][to]` - Max sale date - `filter[catalog]` - Catalog ID - `fields` - Specific fields

Example Request:

```
GET /api/v1.0/datatable-sales?filter[status]=published&sort=-sale_date&range=25
```

Response:

```
{
  "data": [
    {
      "id": "456",
      "label": "Summer Auction 2025",
      "sale_date": "2025-06-15T19:00:00Z",
      "status": "published",
      "catalog": {
        "id": "78",
        "label": "Summer 2025 Catalog"
      },
      "item_count": 250,
      "consignment_count": 15,
      "total_estimates": "250000.00",
      "total_hammer_price": "280000.00",
      "lots_locked": false,
      "venue": "Main Auction House",
      "created": "2025-03-01T10:00:00Z"
    }
  ],
  "count": 12
}
```

Get Single Sale

Endpoint: `GET /api/v1.0/datatable-sales/{id}`

Update Sale

Endpoint: `PATCH /api/v1.0/datatable-sales/{id}`

Request - Lock Lots:

```
{
  "lots_locked": true
}
```

Sales Public

Get publicly available sales for frontend/website.

Endpoint: `GET /api/v1.0/sales-public`

Query Parameters: - `filter[status]` - published - `range` - Items per page - `sort` - Sort order

Example:

```
GET /api/v1.0/sales-public?filter[status]=published&sort=-sale_date
```

Sale Statistics

Get detailed statistics for a sale.

Endpoint: `GET /api/v1.0/sale-stats/{sale_id}`

Query Parameters: - `refresh` - Force refresh (0 or 1)

Response:


```
{
  "data": [{
    "sale_id": "456",
    "total_items": 250,
    "sold_items": 200,
    "unsold_items": 50,
    "sell_through_rate": 80.0,
    "total_hammer_price": "280000.00",
    "total_estimates_low": "200000.00",
    "total_estimates_high": "300000.00",
    "performance_vs_estimate": 112.0,
    "total_buyers": 85,
    "total_bidders": 120,
    "online_bidders": 45,
    "phone_bidders": 15,
    "floor_bidders": 60,
    "average_lot_price": "1400.00",
    "top_lot": {
      "id": "345",
      "lot": "1234",
      "hammer_price": "15000.00"
    }
  }]
}
```


Sale Sessions

Get sale session information.

Endpoint: `GET /api/v1.0/sale-sessions`

Query Parameters: - `filter[sale]` - Sale ID

Example:

`GET /api/v1.0/sale-sessions?filter[sale]=456`

Response:

```
{
  "data": [
    {
      "id": "501",
      "sale": "456",
      "session_date": "2025-06-15T19:00:00Z",
      "session_number": 1,
      "item_range_start": "1",
      "item_range_end": "125",
      "auctioneer": {
        "id": "25",
        "name": "Jane Auctioneer"
      }
    }
  ]
}
```

Create/Update Sale Session

Endpoint: `POST /api/v1.0/sale-sessions`

Request:

```
{
  "sale": "456",
  "session_date": "2025-06-15T19:00:00Z",
  "session_number": 1,
  "auctioneer": "25"
}
```

Endpoint: `PATCH /api/v1.0/sale-sessions/{session_id}`

Sale Session Phone Bidders Card

Get phone bidder cards for printing.

Endpoint: `GET /api/v1.0/sale-session-phone-bidders-card`

Query Parameters: - `filter[session]` - Session ID

Transactions

List Transactions

Get payment transactions for an order.

Endpoint: `GET /api/v1.0/transactions`

Query Parameters: - `filter[order]` - Order ID - `filter[client]` - Client ID - `sort` - Sort order (default: `-created`)

Response:

```
{
  "data": [
    {
      "id": "901",
      "order": "789",
      "transaction_type": "Payment",
      "amount": "500.00",
      "transferred_currency": "USD",
      "transferred_amount": "500.00",
      "bank": {
        "id": "5",
        "name": "Main Bank"
      },
      "bank_account_number": "****1234",
      "credit_card_type": "Visa",
      "credit_card_last_4_digits": "4567",
      "transaction_date": "2025-10-15",
      "cheque_number": null,
      "cheque_date": null,
      "info": "Online payment",
      "cheque_url": null,
      "illustrated_cheque_url": null,
      "created": "2025-10-15T10:30:00Z"
    }
  ]
}
```

Create Transaction

Endpoint: `POST /api/v1.0/transactions`

Request - Credit Card Payment:

```
{
  "order": "789",
  "transaction_type": "Payment",
  "amount": "500.00",
```

```
  "credit_card_type": "Visa",
  "credit_card_last_4_digits": "4567",
  "transaction_date": "2025-10-15",
  "info": "Credit card payment"
}
```

Request - Cheque Payment:

```
{
  "order": "789",
  "transaction_type": "Cheque",
  "amount": "1250.00",
  "bank": "5",
  "bank_account_number": "123456789",
  "cheque_number": "001234",
  "cheque_date": "2025-10-15",
  "transaction_date": "2025-10-15",
  "info": "Cheque payment"
}
```

Request - Wire Transfer:

```
{
  "order": "789",
  "transaction_type": "Wire Transfer",
  "amount": "1250.00",
  "bank": "5",
  "bank_account_number": "987654321",
  "bank_branch": "Main Branch",
  "transaction_date": "2025-10-15",
  "info": "Wire transfer received"
}
```

Update Transaction

Endpoint: `PATCH /api/v1.0/transactions/{id}`

Request:

```
{
  "amount": "550.00",
  "info": "Updated payment amount"
}
```

Delete Transaction

Endpoint: `DELETE /api/v1.0/transactions/{id}`

Generate Cheque

Generate a printable cheque for a transaction.

Endpoint: `POST /api/v1.0/generate-cheque`

Request:

```
{
  "id": "901"
}
```

Response:

```
{
  "data": {
    "transaction_id": "901",
    "cheque_url": "https://backoffice.ddev.site/sites/default/files/cheques/cheque-901.pdf",
    "illustrated_cheque_url": "https://backoffice.ddev.site/sites/default/files/cheques/chec
  }
}
```

Error Response:

```
{
  "title": "Transaction must be of type 'Cheque'",
  "status": 400
}
```

Activities & History

List Activities

Get activity logs for content.

Endpoint: `GET /api/v1.0/activities`

Query Parameters: - `filter[content]` - Content ID (item, order, client, etc.) - `filter[type]` - Activity type ID - `filter[user]` - User ID - `sort` - Sort order (default: `-updated`) - `range` - Items per page

Example:

```
GET /api/v1.0/activities?filter[content]=345&sort=-updated&range=50
```

Response:

```
{
  "data": [
    {
      "id": "7001",
      "type": {
```

```
        "id": "status_change",
        "label": "Status Change"
      },
      "content": {
        "id": "345",
        "type": "item",
        "label": "Item #1234"
      },
      "user": {
        "id": "10",
        "name": "Clerk Name"
      },
      "message": "Status changed from 'available' to 'sold'",
      "details": {
        "old_value": "available",
        "new_value": "sold",
        "hammer_price": "250.00",
        "winner_bidder": "12345"
      },
      "created": "2025-10-15T14:30:00Z",
      "updated": "2025-10-15T14:30:00Z"
    }
  ],
  "count": 25
}
```


Activities List (Simplified)

Get a simplified activity list.

Endpoint: `GET /api/v1.0/activities-list`

Query Parameters: - `filter[user]` - User ID - `filter[date][value][from]` - Start date - `filter[date][value][to]` - End date - `range` - Items per page

Create Activity

Endpoint: `POST /api/v1.0/activities`

Request:


```
{
  "type": "note_added",
  "content": "345",
  "message": "Added note to item",
  "details": {
    "note_text": "Item requires special packing"
```

```
  }
}
```

Notes

List Notes

Get notes for content (items, orders, clients, etc.).

Endpoint: `GET /api/v1.0/notes`

Query Parameters: - `filter[reference]` - Content ID - `sort` - Sort order (default: `-created`)

Example:

```
GET /api/v1.0/notes?filter[reference]=345&sort=-created
```

Response:

```
{
  "data": [
    {
      "id": "6001",
      "reference": "345",
      "note": "Item requires special handling during packing",
      "user": {
        "id": "10",
        "name": "Staff Member"
      },
      "created": "2025-10-10T09:00:00Z"
    }
  ]
}
```

Create Note

Endpoint: `POST /api/v1.0/notes`

Request:

```
{
  "reference": "345",
  "note": "Customer requested additional photos"
}
```

Tasks

List Tasks

Get tasks for content.

Endpoint: `GET /api/v1.0/tasks`

Query Parameters: - `filter[reference]` - Content ID - `filter[status]` - Task status (open, in_progress, completed) - `filter[assigned_to]` - User ID - `sort` - Sort order (default: `created`)

Example:

```
GET /api/v1.0/tasks?filter[reference]=345&filter[status]=open
```

Response:

```
{
  "data": [
    {
      "id": "8001",
      "reference": "345",
      "task_type": {
        "id": "photography",
        "label": "Photography"
      },
      "title": "Take additional photos",
      "description": "Customer requested more detailed images",
      "status": "open",
      "priority": "high",
      "assigned_to": {
        "id": "15",
        "name": "Photographer"
      },
      "due_date": "2025-10-20",
      "created": "2025-10-15T10:00:00Z"
    }
  ]
}
```

Create Task

Endpoint: `POST /api/v1.0/tasks`

Request:

```
{
  "reference": "345",
  "task_type": "photography",
  "title": "Take additional photos",
```

```
  "description": "Customer requested close-ups",
  "assigned_to": "15",
  "due_date": "2025-10-20",
  "priority": "high"
}
```

Update Task

Endpoint: `PATCH /api/v1.0/tasks/{id}`

Request:

```
{
  "status": "completed",
  "completion_notes": "Photos uploaded to item"
}
```

Calls

List Calls

Get call logs for a client.

Endpoint: `GET /api/v1.0/calls`

Query Parameters: - `filter[client]` - Client ID - `sort` - Sort order (default: `-date`)

Example:

`GET /api/v1.0/calls?filter[client]=123&sort=-date`

Response:

```
{
  "data": [
    {
      "id": "9001",
      "client": "123",
      "date": "2025-10-15T11:30:00Z",
      "notes": "Customer inquired about upcoming sales",
      "user": {
        "id": "10",
        "name": "Staff Member"
      },
      "created": "2025-10-15T11:35:00Z"
    }
  ]
}
```

Create Call Log

Endpoint: `POST /api/v1.0/calls`

Request:

```
{
  "client": "123",
  "date": "2025-10-15T11:30:00Z",
  "notes": "Discussed consignment opportunities"
}
```

Shipping

List Shipments

Get shipments for orders.

Endpoint: `GET /api/v1.0/shipments`

Query Parameters: - `filter[order]` - Order ID - `filter[status]` - Shipment status

Create Shipment

Endpoint: `POST /api/v1.0/shipments`

Request:

```
{
  "order": "789",
  "items": ["345", "346", "347"],
  "shipping_method": "FedEx Ground",
  "tracking_number": "123456789012",
  "shipping_cost": "25.00",
  "weight": "2.5",
  "dimensions": {
    "length": "30",
    "width": "20",
    "height": "15",
    "unit": "cm"
  }
}
```

Shipping Areas

Get available shipping areas/zones.

Endpoint: `GET /api/v1.0/shipping-areas`

Response:

```
{
  "data": [
    {
      "id": "10",
      "label": "Local",
      "description": "Within 50 miles",
      "base_rate": "10.00"
    },
    {
      "id": "11",
      "label": "National",
      "description": "Within country",
      "base_rate": "25.00"
    },
    {
      "id": "12",
      "label": "International",
      "description": "Outside country",
      "base_rate": "50.00"
    }
  ]
}
```

Create Shipping Box

Create or update shipping box information.

Endpoint: `POST /api/v1.0/shipping-box`

Request:

```
{
  "order": "789",
  "items": ["345", "346"],
  "box_number": "1",
  "weight": "2.5",
  "dimensions": "30x20x15cm"
}
```

Taxonomy Terms

Categories

Endpoint: `GET /api/v1.0/categories`

Response:

```
{
  "data": [
    {
      "id": "10",
      "label": "Ceramics",
      "parent": null,
      "weight": 0
    },
    {
      "id": "15",
      "label": "Modern Art",
      "parent": null,
      "weight": 1
    }
  ]
}
```

Update Category

Endpoint: `PATCH /api/v1.0/categories/{id}`

Request:

```
{
  "label": "Ceramics & Pottery",
  "weight": 5
}
```

Additional Charges Terms

Endpoint: `GET /api/v1.0/additional_charges`

Response:

```
{
  "data": [
    {
      "id": "5",
      "label": "Shipping Fee",
      "default_amount": "25.00",
      "default_vat": "21.00"
    }
  ]
}
```

Bank Accounts

Endpoint: `GET /api/v1.0/bank_accounts`

Payment Types

Endpoint: `GET /api/v1.0/payment_types`

Bid Steps

Endpoint: `GET /api/v1.0/bid_steps`

Catalogues

Endpoint: `GET /api/v1.0/catalogs`

Catalogue Parts

Endpoint: `GET /api/v1.0/catalogue-parts`

Query Parameters: - `filter[catalog]` - Catalog ID - `sort` - Sort order

Example:

```
GET /api/v1.0/catalogue-parts?filter[catalog]=78&sort=weight
```

Create Catalogue Part

Endpoint: `POST /api/v1.0/catalogue-parts`

Request:

```
{
  "catalog": "78",
  "label": "Section A: Modern Art",
  "description": "Contemporary paintings and sculptures",
  "weight": 1
}
```

Update Catalogue Part

Endpoint: `PATCH /api/v1.0/catalogue-parts/{id}`

Delete Catalogue Part

Endpoint: `DELETE /api/v1.0/catalogue-parts/{id}`

Catalogue Parts Public

Get publicly available catalogue parts.

Endpoint: `GET /api/v1.0/catalogue-parts-public`

Users & Staff

List Staff

Get staff/user list.

Endpoint: `GET /api/v1.0/staff`

Query Parameters: - `filter[role]` - User role - `filter[status]` - User status (active, blocked) - `fields` - Specific fields

Example:

```
GET /api/v1.0/staff?filter[role]=clerk&fields=id,name,email
```

Response:

```
{
  "data": [
    {
      "id": "10",
      "name": "Staff Member",
      "email": "staff@example.com",
      "roles": ["clerk"],
      "status": "active"
    }
  ]
}
```

Get Current User (Me)

Endpoint: `GET /api/v1.0/me`

Response:

```
{
  "data": [{
    "id": "1",
    "name": "John Admin",
    "email": "admin@example.com",
    "roles": ["administrator"],
    "permissions": [...]
  }]
}
```

Update User

Endpoint: `PATCH /api/v1.0/staff/{id}`

Clerk Statistics

Endpoint: `GET /api/v1.0/clerk-stats`

Query Parameters: - `filter[user]` - User ID - `filter[date][value][from]` - Start date - `filter[date][value][to]` - End date

---

Files & Images

Upload File

Upload a file or image.

Endpoint: `POST /api/v1.0/files`

Content-Type: `multipart/form-data`

Form Data: - `file` - The file to upload - `field_name` - Field name (e.g., 'field_images') - `entity_type` - Entity type (e.g., 'node') - `bundle` - Bundle name (e.g., 'item')

Example using curl:

```
curl -X POST \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -F "file=@/path/to/image.jpg" \
  -F "field_name=field_images" \
  -F "entity_type=node" \
  -F "bundle=item" \
  https://backoffice.ddev.site/api/v1.0/files
```

Response:

```
{
  "data": [{
    "id": "1001",
    "fid": "1001",
    "url": "https://backoffice.ddev.site/sites/default/files/item-images/image.jpg",
    "filename": "image.jpg",
    "filesize": "125000",
    "filemime": "image/jpeg"
  }]
}
```

Get File

Endpoint: `GET /api/v1.0/get-file/{fid}`

Response:

```
{
  "data": [{
    "id": "1001",
    "url": "https://backoffice.ddev.site/sites/default/files/item-images/image.jpg",
    "filename": "image.jpg",
    "filesize": "125000"
  }]
}
```

Advanced Queue

Get Queue Item

Get information about a queued background job.

Endpoint: `GET /api/v1.0/advancedqueue_item`

Query Parameters: - `filter[item_id]` - Queue item ID

Bulk Operations

Bulk Dispatcher

Dispatch bulk operations.

Endpoint: `POST /api/v1.0/bulk-dispatcher`

Request:

```
{
  "operation": "print_labels",
  "entity_type": "item",
  "ids": ["345", "346", "347"]
}
```

Common Query Parameters

Pagination

```
?range=25              # Items per page (default: 25)
?page=2                # Page number (0-indexed)
```

Sorting

```
?sort=field_name        # Ascending
?sort=-field_name       # Descending
?sort=-created,id       # Multiple fields
```

Filtering

Simple Filter:

```
?filter[field_name]=value
```

IN Operator (multiple values):

```
?filter[field_name][value][0]=value1
?filter[field_name][value][1]=value2
?filter[field_name][operator]=IN
```

Range Filter:

```
?filter[field_name][value][from]=100
?filter[field_name][value][to]=1000
```

Date Range:

```
?filter[created][value][from]=2025-01-01
?filter[created][value][to]=2025-12-31
```

Regex Search:

```
?filter[field_name][value]=search.*term
?filter[field_name][operator]=REGEXP
```

Field Selection

```
?fields=id,label,status,created
```

Global Search

```
?global_search=search+term
```

Error Responses

Standard Error Format

```
{
  "title": "Validation error",
  "status": 400,
  "detail": "The field 'email' is required",
  "errors": {
    "email": ["This field is required"]
```

```
  }
}
```

Common HTTP Status Codes

- `200 OK` - Request successful
- `201 Created` - Resource created successfully
- `204 No Content` - Success with no response body (e.g., DELETE)
- `400 Bad Request` - Invalid request data
- `401 Unauthorized` - Authentication required or failed
- `403 Forbidden` - Insufficient permissions
- `404 Not Found` - Resource not found
- `422 Unprocessable Entity` - Validation error
- `500 Internal Server Error` - Server error

Error Examples

Missing Required Field:

```
{
  "title": "The field 'email' is required",
  "status": 400
}
```

Invalid Data Format:

```
{
  "title": "Invalid email format",
  "status": 422,
  "errors": {
    "email": ["Please enter a valid email address"]
  }
}
```

Permission Denied:

```
{
  "title": "Access denied",
  "status": 403,
  "detail": "You do not have permission to perform this action"
}
```

Resource Not Found:

```
{
  "title": "Resource not found",
  "status": 404,
  "detail": "Order with ID 999 does not exist"
}
```

Rate Limiting

API requests may be rate-limited based on IP address or access token.

Response Headers: - `X-RateLimit-Limit` - Maximum requests per window - `X-RateLimit-Remaining` - Remaining requests in current window - `X-RateLimit-Reset` - Unix timestamp when limit resets

Rate Limit Exceeded:

```
{
  "title": "Rate limit exceeded",
  "status": 429,
  "detail": "Too many requests. Please try again later.",
  "retry_after": 60
}
```

Data Types & Formats

Dates

All dates are in ISO 8601 format (UTC):

`2025-10-15T14:30:00Z`

For date-only fields:

`2025-10-15`

Currency

Currency amounts are strings to preserve precision:

```
{
  "amount": "1250.50"
}
```

Formatted amounts include currency symbol:

```
{
  "amount": "1250.50",
  "formatted": "$1,250.50"
}
```

Booleans

Boolean values can be: - `true` / `false` - `1` / `0` - `"1"` / `"0"`

NULL Values

- Missing optional fields may be omitted
- Explicitly null fields: `null`
- Empty strings: `""`

Versioning

The API uses URL-based versioning. Current version: `v1.0`

All endpoints include the version in the path:

`/api/v1.0/resource`

Best Practices

Efficient Filtering

Combine filters to reduce payload:

```
GET /api/v1.0/orders?filter[type]=bidder&filter[status]=new&filter[sale]=456&range=25
```

Field Selection

Request only needed fields:

```
GET /api/v1.0/datatable-clients?fields=id,first_name,last_name,email
```

Batch Operations

Use bulk endpoints for multiple updates:

```
POST /api/v1.0/items-bulk
```

Pagination

Use reasonable page sizes:

```
GET /api/v1.0/orders?range=50    # Good
GET /api/v1.0/orders?range=1000  # Avoid
```

Error Handling

Always check HTTP status codes and handle errors appropriately.

Caching

- Use ETags when available
- Cache reference data (categories, payment types, etc.)

Support

For API support, questions, or bug reports, contact the development team.

Last Updated: October 30, 2025 **API Version:** 1.0