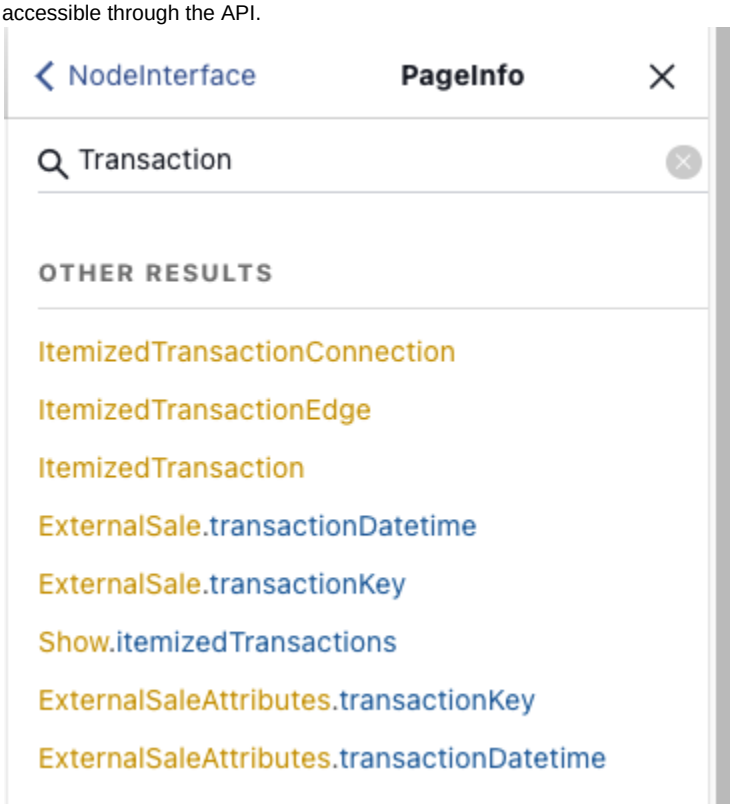
Transaction Read Specific Examples

- Query Definitions:
- Pagnation (Paging)
- getTransactions
- getShows
- getTransactionsForShow
  - Series Execution Process
- getShows Limitation to Wide Date Range

The query is useful in displaying the number of Transactions per Show…….

Query Definitions:

Docs: section for GraphiQL located on the upper right side of the api-browser. Use the search field to learn more about the data that is accessible through the API.



Pagnation (Paging)

The process of separating content into discrete pages.

Paginating through collections *nested* within paginated collections is not recommended

A paged collection nested in another paged collection that returns an object that has more than one page of data cannot be properly paged. The nested collection's parent should be queried directly so that the collection that requires paging can be properly paged at the top level.

Example: The sample query below selects 4 collections, three of them nested:

- organization
  - accounts collection Can be paged because it is the first collection
    - tours collection nested, but will there will only be 1 tour for Venues
      - shows collection may need paging depending on arguments
        - itemizedTransactions collection most likely needs paging - use seperate query

To retrieve all the transactions for a particular show make a separate query for only that show and page through the transactions

- show(uuid: $showUuid)
  - itemizedTransactions collection

getTransactions

Query

```
query getTransactions($pageSize: Int!, $startDate: Date!, $endDate:
Date!) {
  organization {
    name
    accounts(first: $pageSize) {
      pageInfo {
        endCursor
        hasNextPage
      }
      nodes {
        name
        uuid
        externalId
        tours(first: 1) {
          nodes {
            shows(first: $pageSize, sortDirection: DESCENDING,
showsOverlap: {start: $startDate, end: $endDate}) {
              pageInfo {
                hasNextPage
                endCursor
              }
              nodes {
                uuid
                externalId
                state
                showDate
                showEndDate
                currencyFormat {
                  code
                }
```

```
                itemizedTransactions(first: $pageSize) {
                  pageInfo {
                    hasNextPage
                    endCursor
                  }
                  nodes {
                    artistName
                    cardholderName
                    device
                    itemName
                    unitPriceAmount
                    size
                    refundedQuantity
                    soldQuantity
                    netSoldAmount
                    orderId
                    orderType
                    paymentTimestamp
                    staffName
                    standName
                    tenderType
                    totalDiscountAmount
                    totalRefundedAmount
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Variables

```
{
  "pageSize": 10,
  "startDate": "2019-05-01",
  "endDate": "2019-09-30"
}
```

Shows older than January 1, 2019 may have a null orderId

Response

```
{
  "data": {
    "organization": {
      "name": "My Org",
      "accounts": {
        "pageInfo": {
          "endCursor": "MTA=",
          "hasNextPage": true
        },
        "nodes": [
          {
            "name": "My Amphitheater",
            "uuid": "acct_XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
            "externalId": null,
            "tours": {
              "nodes": [
                {
                  "shows": {
                    "pageInfo": {
                      "hasNextPage": true,
                      "endCursor": "MTA="
                    },
                    "nodes": [
                      {
                        "uuid": "show_XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX",
                        "externalId": null,
                        "state": "DONE",
                        "showDate": "2019-10-18",
                        "showEndDate": "2019-10-18",
                        "currencyFormat": {
                          "code": "USD"
                        },
                        "itemizedTransactions": {
                          "pageInfo": {
                            "hasNextPage": true,
                            "endCursor": "MTA4NzU1NThfOTA2MjEy"
                          },
                          "nodes": [
                            {
                              "artistName": "Artist Name",
                              "cardholderName": "EIVGhNkLyUL34oB",
                              "device": "atVenu Register YYY",
                              "itemName": "Shirt",
                              "unitPriceAmount": "0.35e2",
                              "size": "M",
                              "refundedQuantity": 0,
                              "soldQuantity": 1,
                              "netSoldAmount": "0.35e2",
                              "orderId": "ZZZZZZ",
```

```
                                "orderType": "On Site",
                                "paymentTimestamp": "2019-10-19T04:01:
00.736Z",
                                "staffName": null,
                                "standName": null,
                                "tenderType": "Credit card",
                                "totalDiscountAmount": "0.0",
                                "totalRefundedAmount": "0.0"
                            },
                            ...
                        ]
                    }
                },
                ...
            ]
        }
    },
    ...
  ]
 }
}
```


getShows

Fetching all transactions for every show in every account of an organization.

The query is limited to one night of shows.

If you are interested in last night's shows - use yesterday's date for the `start` and `end` in the example provided below.

Assumptions

- The organization has many accounts (more than the page size)
- Each account only has one tour each (less than the page size)
- We are only interested in last night (less than the page size)
- Each show has many transactions (more than the page size)

Query

Gather the list of yesterday's shows from all of the accounts under an organization.

```
query getShows($pageSize: Int, $range: DateRange, $accountsCursor:
String) {
  organization {
    accounts(first: $pageSize, after: $accountsCursor) {
      pageInfo {
        hasNextPage
        endCursor
      }
      accountNodes: nodes {
        name
        uuid
        externalId
        tours(first: 1) {
          tourNodes: nodes{
            uuid
            shows(first: $pageSize, showsOverlap: $range) {
              pageInfo {
                hasNextPage
                endCursor
              }
              showNodes: nodes {
                showDate
                showEndDate
                uuid
                externalId
              }
            }
          }
        }
      }
    }
  }
}
```

Variables

In this example, *yesterday* is June 15, 2019

```
{
  "pageSize": 20,
  "range": {"start": "2019-06-15", "end": "2019-06-15"},
  "accountsCursor": null
}
```

Response

```
{
  "data": {
    "organization": {
      "accounts": {
        "pageInfo": {
          "hasNextPage": true,
          "endCursor": "MjA="
        },
        "accountNodes": [
          # ...
          {
            "name": "Amphitheater A",
            "uuid": "acct_AAAAA-AAAA-AAAA-AAAA-AAAAAAAA",
            "externalId": "CustomVenueID-A",
            "tours": {
              "tourNodes": [
                {
                  "uuid": "tour_AAAAA-AAAA-AAAA-AAAA-AAAAAAAA",
                  "shows": {
                    "pageInfo": {
                      "hasNextPage": false,
                      "endCursor": "MQ=="
                    },
                    "showNodes": [
                      {
                        "showDate": "2019-06-15",
                        "showEndDate": "2019-06-15",
                        "uuid": "show_AAAAA-AAAA-AAAA-AAAA-000000001",
                        "externalId": "CustomShowID-A-1"
                      }
                    ]
                  }
                }
              ]
            }
          },
          {
            "name": "Amphitheater B",
            "uuid": "acct_BBBBB-BBBB-BBBB-BBBB-BBBBBBBB",
            "externalId": "CustomVenueID-B",
            "tours": {
              "tourNodes": [
                {
                  "uuid": "tour_BBBBB-BBBB-BBBB-BBBB-BBBBBBBB",
                  "shows": {
                    "pageInfo": {
                      "hasNextPage": false,
                      "endCursor": null
                    },
                    "showNodes": []
```

```
                    }
                  }
                ]
              }
            },
            {
              "name": "Amphitheater C",
              "uuid": "acct_CCCCC-CCCC-CCCC-CCCC-CCCCCCCC",
              "externalId": "CustomVenueID-C",
              "tours": {
                "tourNodes": [
                  {
                    "uuid": "tour_6400ad7d-41cc-46c0-837f-ed48d673507b",
                    "shows": {
                      "pageInfo": {
                        "hasNextPage": false,
                        "endCursor": "MQ=="
                      },
                      "showNodes": [
                        {
                          "showDate": "2019-06-14",
                          "showEndDate": "2019-06-15",
                          "uuid": "show_CCCCC-CCCC-CCCC-CCCC-00000001",
                          "externalId": "CustomShowID-C-1"
                        },
                        {
                          "showDate": "2019-06-15",
                          "showEndDate": "2019-06-16",
                          "uuid": "show_CCCCC-CCCC-CCCC-CCCC-00000002",
                          "externalId": "CustomShowID-C-2"
                        }
                      ]
                    }
                  }
                ]
              }
            },
            # ...
          ]
        }
      }
    }
  }
```

getTransactionsForShow

Query

Lists itemized sales Transactions per Show.

```
query getTransactionsForShow($pageSize: Int, $showUuid: UUID!,
$transactionsCursor: String) {
  showNode: node(uuid: $showUuid) {
    ...on Show {
      itemizedTransactions(first: $pageSize, after:
$transactionsCursor) {
        pageInfo {
          hasNextPage
          endCursor
        }
        transactionNodes: nodes {
          orderId
          # ... additional fields
        }
      }
    }
  }
}
```

Variables

```
{
  "pageSize": 200,
  "showUuid": "show_AAAAA-AAAA-AAAA-AAAA-000000001",
  "transactionsCursor": null
}
```

Response

```
{
  "data": {
    "showNode": {
      "itemizedTransactions": {
        "pageInfo": {
          "hasNextPage": true,
          "endCursor": "ODQ1MTc0N182OTY1MDE="
        },
        "transactionNodes": [
          {
            "orderId": "0ec2d3",
            # ...
          },
          {
            "orderId": "c01381",
            # ...
          },
          {
            "orderId": "c01381",
            # ...
          },
          # ...
        ]
      }
    }
  }
}
```

Series Execution Process

- execute `getShows` to retrieve the list of accounts and shows
- Do
  - for each account in `data.organization.accounts.accountNodes`
  - Do
    - for the first tour in `account.tours.tourNodes`
      - for each show in `tour.shows.showNodes`
        - set `showUuid` in getTransactionsForShow variables to `show.uuid`
        - Do
          - execute `getTransactionsForShow` to get the list of transactions for a show
          - process the list of transactions contained in `data.showNode.itemizedTransactions.transactionNodes`
          - set `transactionsCursor` in getTransactionsForShow variables to `data.showNode.pageInfo.endCursor`
        - While `data.showNode.itemizedTransactions.pageInfo.hasNextPage === true`
  - set `accountsCursor` in getShows variables to `data.organization.accounts.pageInfo.endCursor`
- While `data.organization.accounts.pageInfo.hasNextPage`

getShows Limitation to Wide Date Range

Show queries are limited to a maximum page size of 20 Accounts per query. Review the Process below if you are interested in the shows across a wide date range.

Assumptions

Same as above except:

- Many Shows per Tour (more than the page size)

Process

- Use the process above with the assumption that most Accounts will have less shows than the page size for a given date range. Ensure for each Tour that `` `tour.shows.pageInfo.hasNextPage` `` === `false`. If it is true, then process that Tour separately by querying the Tour node directly.

Alternatively:

- If the process above is of the assumption that there will be more Shows than the page size than query only Tours initially for each Account. And then for each Tour, query the Shows, followed by a query for each Show's itemized Transactions.