# Transactions

## List Transaction

To list transactions on the system use the following operation

```
GET https://<server_address>/api/tool/Transaction/
?hierarchy=[hierarchy]
&format=json
&summary=true
```

The following query parameter illustrates how a second page of 50 transactions in the transaction log is requested.

```
skip=50
&limit=50
&hierarchy=[hierarchy]
&format=json
&summary=true
&direction=desc
```

```
&order_by=submitted_time
```

For further information on the query parameters refer to API Parameters above.

The synchronous response contains:

- pagination information
- meta data specifying the summary attributes of the transaction log view
- resources containing a list of the transactions in the transaction log

# Get Instance Transactions

The status of a specific transaction can be retrieved by using a GET call to /tool/Transaction for a specific transaction PKID. The PKID, also known as the transaction ID, or transaction_id is available in the synchronous response to an asynchronous mutator transaction.

For example, if the transaction_id in the response is [pkid], then the transaction can be polled with:

```
GET https://<server_address>/api/tool/Transaction/[pkid]
```

# Poll Transactions

It is recommended to use asynchronous transaction call back mechanism described in “Asynchronous Mutator Transaction Status Callback”. If this can not be used, a consumer of the Cisco Unified Communications Domain Manager 10.x/11.5(x) API can also use this polling mechanism to poll the status of individual transactions using the poll action of the transaction tool. A user interface that allows a user to monitor the progress of a given transaction can use the following method to retrieve the status of a given transaction:

```
GET /api/tool/Transaction/[pkid]/poll/?format=json
```

The response contains essential status of the transaction, for example:

```
{
[pkid]: {
status: "Success",
href: "/api/tool/Transaction/[pkid]",
description: "Name:RDP-auser1857 Description:RD for auser1857"
}
}
```

# Replay Transactions

Transactions that have failed can, under certain circumstances, be replayed. This means that the transaction is re-submitted with the original request parameters. This is done by specifying the PKID of the transaction

```
GET https://<server_address>/api/tool/Transaction/[pkid]/replay/
```

The transaction current operation replays the transaction and the result returns the list view of the transaction log.

# Edit and Reply Transactions

Transactions can be edited and then replayed under certain circumstances. This means that the transaction is re-submitted with the updated request parameters. This is done by specifying the PKID of the transaction:

```
GET https://<server_address>/api/tool/Transaction/[pkid]/edit-replay/
```

The transaction current operation edit and then replays the transaction and the result returns the list view of the transaction log.

# Sub Transactions

The sub-transactions of a transaction with PKID can be retrieved by submitting the following URI

```
GET https://<server_address>/api/tool/Transaction/[pkid]/sub_transaction/
```

# Log Transactions

The log messages of a transaction with PKID can be retrieved by submitting the following URI

```
GET https://<server_address>/api/tool/Transaction/[pkid]/log/
```

# Transaction Choices

Use the available URL endpoint and parameter to list the transaction actions as they may be shown in the transaction log.

The API call to get the list of transaction actions uses the parameter and value: field=action, for example:

```
GET api/tool/Transaction/choices/?
    field=action&
    hierarchy=[hierarchy]&
    format=json
```

The output shows the list of transaction actions:

```
[
    {
    "value": "Auto Migrate Base Customer Dat",
    "title": "Auto Migrate Base Customer Dat"
    },
    {
    "value": "Auto Migrate Base Provider",
    "title": "Auto Migrate Base Provider"
    },
    {
    "value": "Auto Migrate Base Site Dat",
    "title": "Auto Migrate Base Site Dat"
    },
    {
    "value": "Auto Migrate Dial Plan",
    "title": "Auto Migrate Dial Plan"
    },
    {
    "value": "Auto Migrate Feature Subscriber Phone Cft",
    "title": "Auto Migrate Feature Subscriber Phone Cft"
    },
    {
    "value": "Auto Migrate Hotdial Data",
    "title": "Auto Migrate Hotdial Data"
    },
    {
    "value": "Auto Migrate Init Ippbx",
    "title": "Auto Migrate Init Ippbx"
    },
    {
    "value": "Auto Migrate Internal Number Inventory",
    "title": "Auto Migrate Internal Number Inventory"
},
...
```

# Transaction Filters

In addition to the filter parameters that can be applied to transactions as indicated in the topic on AP Parameters, transactions in particular can be filtered:

- by the following values for the URL parameter `filter_field`:
    - Transaction ID: `id`
    - Start or end submitted time: `submitted_time`
    - The transaction message: `message`
- by also listing sub transactions using the URL parameter and value `subtransactions=true`. By default, sub transactions are not listed, in other words, the value is `false`.
- To carry out a filter on sub-transactions of a parent transaction, the `/sub-transactions/` endpoint is added to the GET request:

  `/api/tool/Transaction/[parent-pkid]/sub-transactions/`
- To carry out a filter on transaction logs of a parent transaction, the `/logs/` endpoint is added to the GET request:

  `/api/tool/Transaction/[parent-pkid]/log/`

  The transaction filters do not apply to logs.

The parameters can have the `filter_condition` values:

- `eq` (equals)
- `ne` (not equals)
- `gt` (greater than)
- `gte` (greater than or equals)
- `lt` (less than)
- `lte` (less than or equals)

The date-time is a `filter_text` value for `filter_field=submitted_time`.

The format follows RFC3339: `[https://tools.ietf.org/html/rfc3339]` and is *YYYY-MMDDTHH:MM:SS.fZ*, where:

- “T” is the time separator and the character should be added.

- “Z” indicates UTC time and the character should be added.
- “f” represents the decimal fraction of a second and the character should not be added. The specification of the decimal fraction is optional.

For example:

June 29 2016 14 hours 41 minutes 0.01 seconds UTC, is:

```
2016-06-29T14:41:00.01Z
```

To filter for transactions after this date-time, the API call is:

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=submitted_time
    &filter_text=2016-06-29T14:41:00.01Z
    &filter_condition=gt
```

To filter between transaction IDs or times, two parameter sets are needed.

For example:

- To filter transaction IDs between 12000 and 13000:
  ```
  GET api/tool/Transaction/?
      hierarchy=[hierarchy]
      &format=json
      &filter_field=id
      &filter_text=12000
      &filter_condition=gt
      &filter_field=id
      &filter_text=13000
      &filter_condition=lt
  ```
- To filter transactions between June 29 2016 14 hours 41 minutes UTC and June 29 2016 15 hours 41 minutes UTC (no fraction of a second in the example):
  ```
  GET api/tool/Transaction/?
      hierarchy=[hierarchy]
      &format=json
      &filter_field=submitted_time
      &filter_text=2016-06-29T14:41:00Z
      &filter_condition=gt
      &filter_field=submitted_time
      &filter_text=2016-06-29T15:41:00Z
      &filter_condition=lt
  ```

If the upper or lower bound in the filter are not available, the transactions with values between the filte values and the bound are returned.

When the URL parameter `subtransactions=true` is used, the `data` object in the JSON API response shows:

- a parent transaction has: `parent: null`
- a sub transaction has: `parent: <pkid>`, where `<pkid>` is the value of the parent attribute `pkid`.

The example snippets below show the values of `parent`:

```
data: {
    username: "system",
    status: "Success",
    description: "",
    parent: null,
    pkid: "01a559c5-e77f-40e7-8403-683d7204d1e1",
    friendly_status: "Success",
    detail: "HcsLdapSyncSchedule--1",
    action: "Execute Schedule",
    href: "/api/tool/Transaction/01a559c5-e77f-40e7-8403-683d7204d1e1/",
    txn_seq_id: "17693",
```

```
    data_type_: "tool/Transaction",
    message: "",
    submitted_time: "2016-07-14T12:13:41.758000Z"
}
data: {
    username: "system",
    status: "Success",
    description: "",
    parent: "f4daa234-590d-4002-a3b0-8c329c583d1d",
    pkid: "019f44a3-df6e-4e4f-86f3-a09a6b91e482",
    friendly_status: "Success",
    detail: "10.120.2.221",
    action: "Import Ldap",
    href: "/api/tool/Transaction/019f44a3-df6e-4e4f-86f3-a09a6b91e482/",
    txn_seq_id: "17695",
    data_type_: "tool/Transaction",
    message: "models completed.",
    submitted_time: "2016-07-14T12:13:43.075000Z"
}
```

- In the case of a transaction error, the `message` attribute value will show the corresponding error message. If a custom message was defined in a provisioning workflow, and the response is the result of the workflow, the value will be the custom message.

  To filter transaction messages, the parameter `filter_field=message` is used, with at least one of the following additional filter criteria:

  - a date range of maximum 7 days using `submitted_time`
  - an additional `filter_field`, with one of the conditions:
    - `filter_condition=contains`
    - `filter_condition=startswith`

    Also required is the case sensitive parameter:

    - `ignore_case`

    **Note** Note that a filter is by default case insensitive. If the case is explicitly set, then it must be added to each filter parameter group to ensure proper parameter grouping.

The additional criteria do not apply to sub-transaction message filters, because the `[parent-pkid]` in the URL serves as an additional filter.

The example below is a message filter that contains “Invalid business key”, by date range:

```
GET api/tool/Transaction/?
    hierarchy=[hierarchy]
    &format=json
    &filter_field=submitted_time
    &filter_text=2016-06-25T14:41:00Z
    &filter_condition=gt
    &ignore_case=false
    &filter_field=submitted_time
    &filter_text=2016-06-29T15:41:00Z
    &filter_condition=lt
    &ignore_case=false
    &filter_field=message
    &filter_text=Invalid%20business%20key
    &filter_condition=contains
    &ignore_case=false
```