Expense SFTP and API Integration Guide

Modified: July 8th, 2025

Table of Contents

Overview

Navan can integrate to various ERPs/Accounting Systems in multiple ways.
Depending on the complexity of ERP Setup at clients' systems, different options can be used to integrate seamlessly with Navan.

There are mainly three types of integration to any Accounting System/ERP:

- Push(Direct) integration from Navan to Clients ERP/Accounting Systems.
- Pull Integration from Client-side Middleware connecting to Navan.
- Customers Manually download files from Navan Platform and upload manually to ERP System.

Push(Direct) Integrations:

- **Netsuite** Direct Integration
- **QBO** Direct Integration
- **XERO** Direct Integration
- **Sage Intacct** Direct Integration

Pull Integrations:

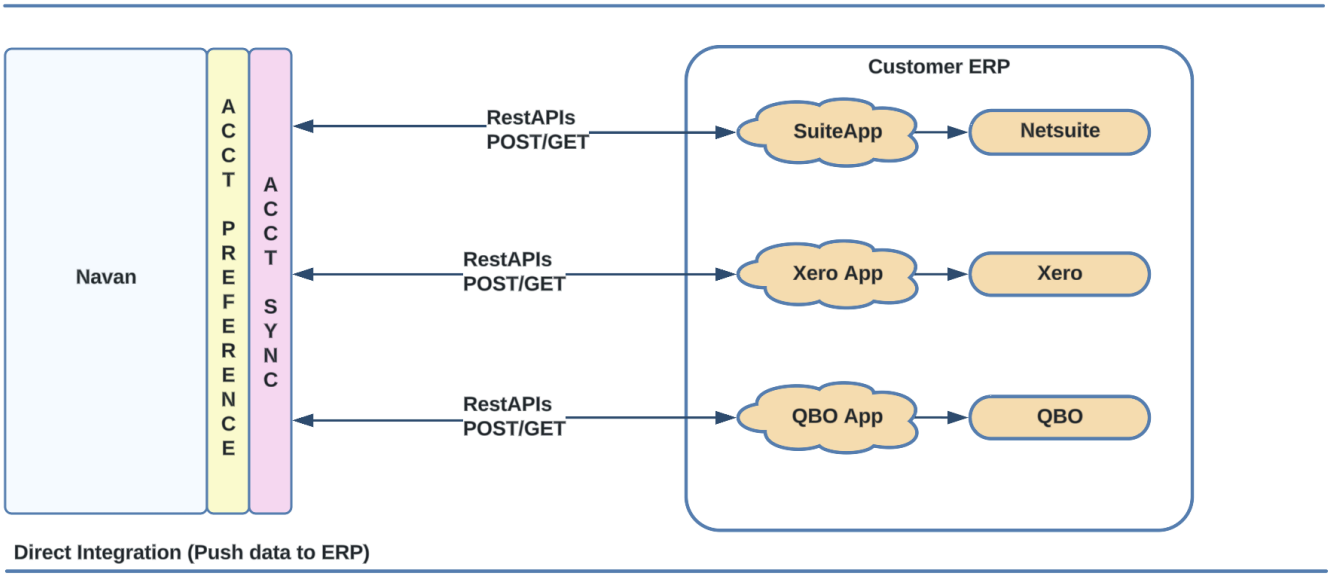
There are 3 ways to pull data from Navan:

- SFTP connection to Navan Server and get Transactions/Statement files
- RestAPI (Transaction Partner API)
- Manual (Download from Navan Dashboard - Platform)

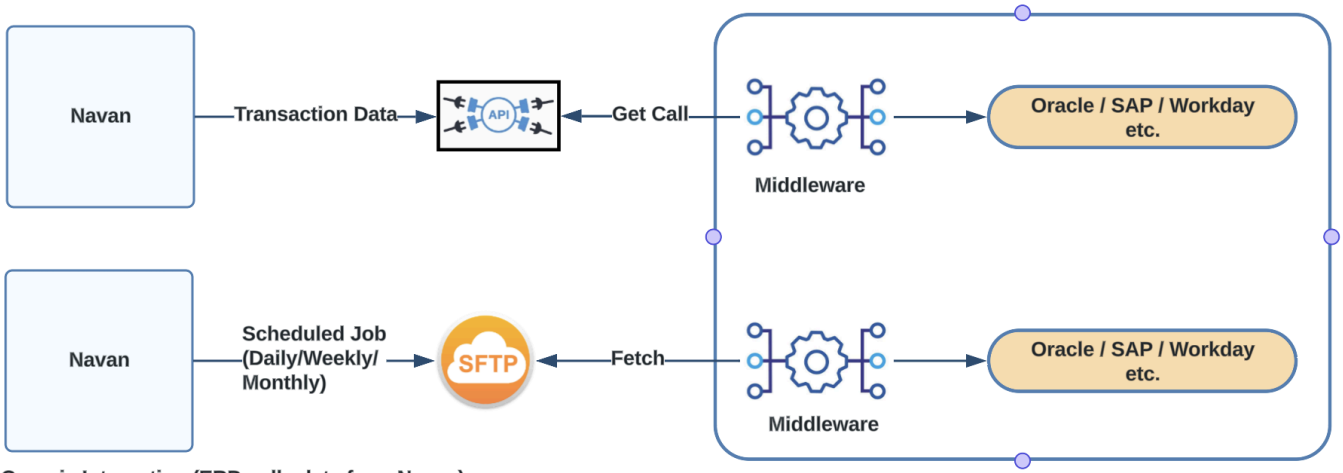
Most of the Enterprise customers will leverage **SFTP** integration as it gives more flexibility to the build their own customized flow into the ERP systems. Direct integration to ERPs like **Netsuite** and **QBO** are very straightforward integrations. For complex flows either **SFTP** or **RestAPI** is a better approach.

SFTP	RestAPI
Schedule Daily, Weekly or Monthly	Only Daily transaction data
Async process	Sync process
PGP Encrypted	HTTPS
Statements Available	No Statements API
Schedule Reports Self Serve Page to Monitor and reprocess	-NA-
File data in CSV format	API response data in JSON format

High Level Architecture



Direct Integration (Push data to ERP)



Generic Integration (ERP pulls data from Navan)

SFTP

Data security Details

- Customers' data access is done via SFTP interface. Access to the customer is provided using AWS transfer family server (AWS fully managed service - more details here). Key aspects of the AWS transfer family are
  - **Compliant with numerous certifications** (e.g., PCI DSS, HIPAA), whereas maintaining these certifications with traditional SFTP servers can be burdensome.
  - **Uses built-in encryption at rest** (S3 native encryption)
  - **Leverages AWS Key Management System** (KMS) for key management to ensure proper encryption. It does not rely on traditional SFTP key management which is manual and error prone
  - **Provides high availability and reliability** as it relies on AWS multi region and availability zone architecture, unlike traditional SFTP that requires manual setup and maintenance to ensure them.
- Customer authentication
  - Navan assigns each customer a unique customer ID to use for login
  - Each customer provides their public SSH key as part of configuration setup of HRIS
  - Navan provides a host key to all customers to trust on first connection
  - When customers sign in, they provide their user ID and private key to authenticate against the above user ID and public key
- Customer data
  - Customer data is stored in AWS S3 bucket, encrypted and private
  - Each customer's data is stored in a separate export folder. Access to these folders is controlled via IAM roles that are associated with the SFTP access users (SFTP authentication is done via RSA private/public key only).
  - Folder separation is enforced via aws IAM access policy. This makes sure that customers cannot access data that is not part of their company's data and that does not belong to them

The system also supports **PGP encryption** on every transaction file using the PGP public key provided by the customer. This provides another layer of security at the file level. Data can only be accessed with the customer's private PGP key.

Overview SFTP files and folders

SFTP folders are created depending on what kind of data is extracted on a **DAILY/WEEKLY/MONTHLY** basis.

Example:

- transactions/daily **or** transactions/weekly **or** transactions/monthly
- manual_transactions/daily **or** manual_transactions/weekly **or** manual_transactions/monthly
- connect/daily **or** connect/weekly **or** connect/monthly
- repayments/daily **or** repayments/weekly **or** repayments/monthly
- Statements (This folder holds all consolidated statements)

SFTP transactions folder holds all Navan Credit card transactions data.

- **ACTIVITY_TYPE**: Purchase, Refund
- **POSTED_DATE**: This date is a credit card charge_date. Hence you see only Posted credit card transactions **(irrespective of whether its APPROVED or REJECTED or FLAGGED**) in this folder.

SFTP manual_transactions Folder holds all manual transactions (both **Navan reimbursed** as well as customer reimbursed (**PAYROLL**))

- **ACTIVITY_TYPE**: Manual Transaction
- **REIMBUSEMENT_METHOD**: **ACH** or **PAYROLL**
- **POSTED_DATE** and **REIMBUSEMENT_DATE** are the same and for **PAYROLL** Transactions **POSTED_DATE** will be null.
- Only **Approved/Reimbursed** data are in these files. (**APPROVAL_STATUS** = **APPROVED**)

SFTP repayments Folder holds all repayments done by employees for their personal expenses.

- **ACTIVITY_TYPE**: repayment

SFTP connect Folder holds all other card programs data (AMEX, BOFA, CITI etc.)

- **ACTIVITY_TYPE**: Purchase, Refund
- **POSTED_DATE**: This date is a credit card charge date. Hence you see only posted connect card transactions **(irrespective of whether its APPROVED or REJECTED or FLAGGED)** in this folder.

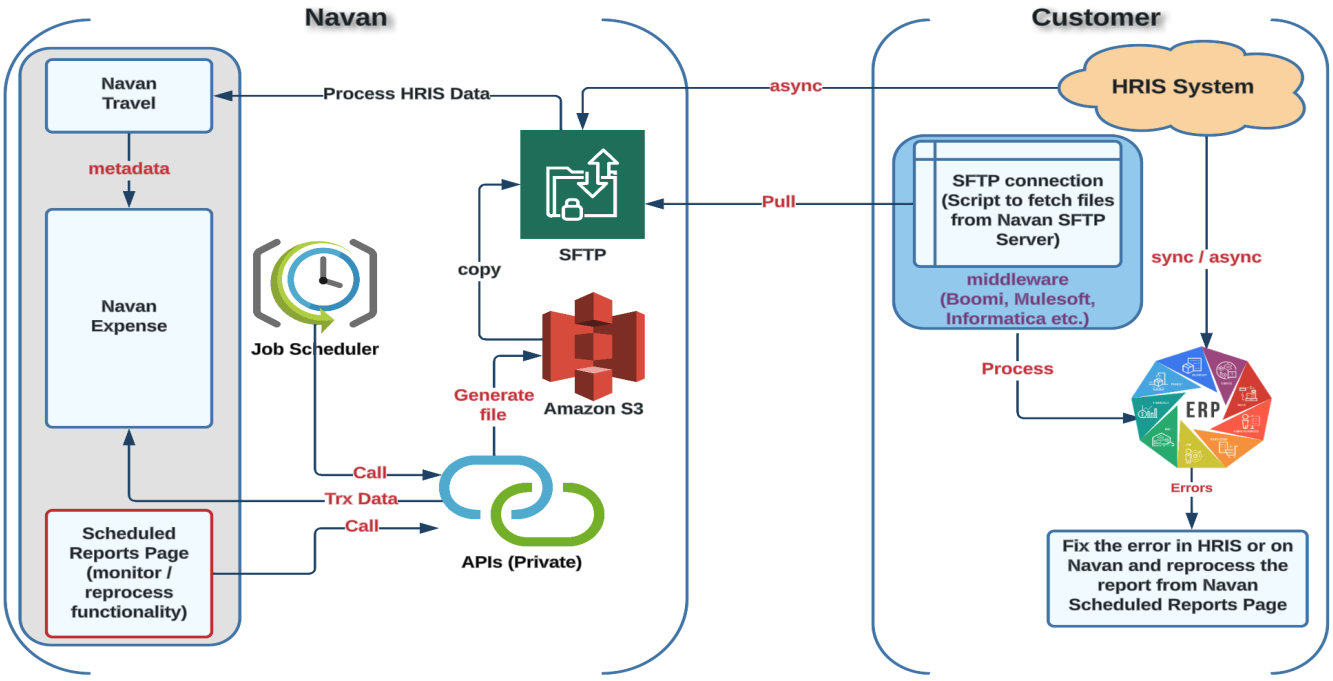
SFTP statements folder holds all legal entities statements (consolidated)

- Statements hold only transactions that are Navan reimbursed (Navan cards and Navan payments). **PAYROLL** and **CONNECT** transactions are not included in statements.

Configuration Details

Item	Description
Navan SFTP Server Details	**EndPoint:** **sftp://sftp-liquid.tripactions.com** **Port:** **22**
Authorization (Required)	Navan uses Public/Private RSA SSH Keys for authentication. **Please provide your server's public RSA key to Navan CSM.**
Directory Structure	**/transactions/monthly/** **/transactions/weekly/** **/transactions/daily/** **/manual_transactions/monthly/** **/manual_transactions/weekly/** **/manual_transactions/daily/** **/repayments/monthly/** **/repayments/weekly/** **/repayments/daily/** **/connect/daily/** **/connect/weekly/** **/connect/monthly/** **/statements/** **/statements/receipts/ (optional)** **File name format for transactions:** <transaction_type>_<reportDate YYYY-MM-DD>_<RunDate YYYY_MM_DD_HH24_MI_SS>.csv.gpg (Ex: transactions_2024-03-28_2024_03_29_01_00_16.csv.gpg manual_transactions_2024-03-28_2024_03_29_01_00_15.csv.gpg repayments_2024-03-28_2024_03_29_01_00_26.csv.gpg) **File name format for statements:** Depending on the Statement generation configuration (**Weekly/Semi Monthly/Monthly**) the file naming convention is: **Monthly File** : statement_<month start date(YYYY_MM_DD)>.csv.gpg (Ex: statement_2024_12_01.csv.gpg) **Semi Monthly File**: statement_<First half of month start date(YYYY_MM_DD)>.csv.gpg and statement_<second half of month start date(YYYY_MM_DD)>.csv.gpg (Ex: statement_2024_12_01.csv.gpg and statement_2024_12_16.csv.gpg) **Weekly Files**: statement_<week start date (YYYY_MM_DD)>.csv.gpg (Tue - Mon)(Ex: statement_2024_07_09.csv.gpg) ***Note: All files are consolidated across all legal entities.*** **Depending on the frequency set in the configuration, respective directories are created to access.**
Encryption (Required)	Navan will need a PGP public key to encrypt the file. - **Client to provide PGP public key to Navan CSM** - File will be a .gpg - Client have to decrypt the file using PGP private key

Flow Diagram



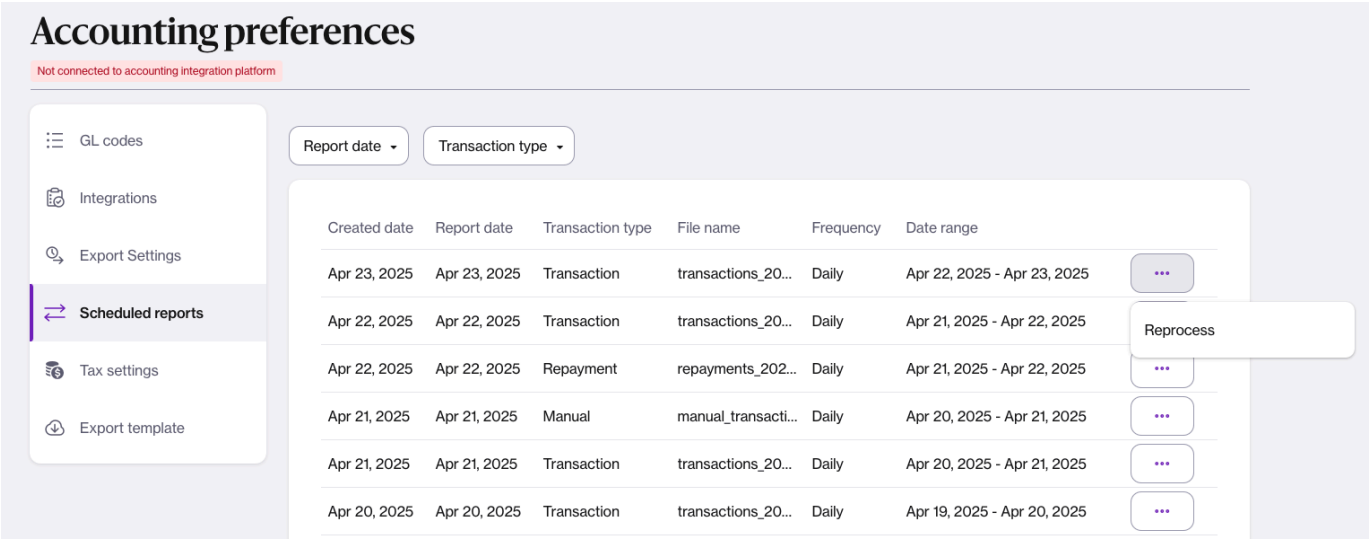
Frequency of transaction/Expense reports can be configured to Daily/weekly/ Monthly

SFTP Scheduled Reports Page

Expense Admin -> Company -> Accounting Preference -> Scheduled Reports (sidebar) Page

This page displays all the files (**TRANSACTIONS, MANUAL_TRANSACTIONS, CONNECT and REPAYMENTS**) that are generated as the schedule (**DAILY, WEEKLY or MONTHLY**).

To reprocess a file, click on the … button and select **“reprocess”.**



Note:

- Daily Schedule runs at 1am UTC time for the previous day.
- Weekly Schedule runs at 1am UTC every Sunday for the previous week (monday to sunday)
- Monthly Schedule runs at 1am UTC on the first day of the month for the previous month.

Transactions Partner API

Navan transactions data is available via API for customers who wish to transfer this data to their ERP/Accounting Systems
(Customer has to be enabled to access the Transaction API, this is a manual process that will be done by Navan support team) .

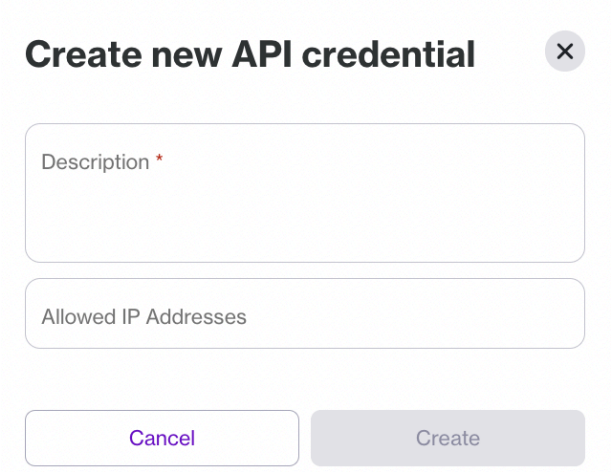
Customer onboarding

Prior to following the below steps, any customer who wants to use the “Expense API” is to request the Navan team to enable the public API for that client. It is also **mandatory** that the customer should be enabled for Navan Expense services.

Step 1: Generate API credentials

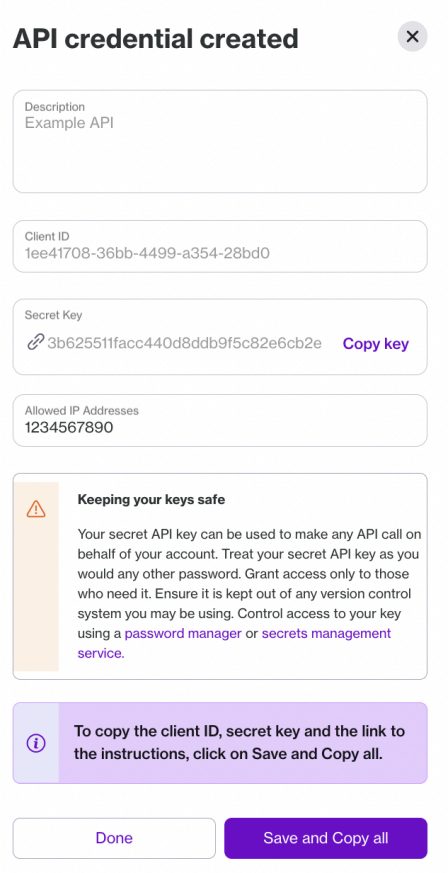
Creating new API credentials

1. Log in to your Navan account on web
2. From the main dropdown menu on the left, select **Admin**, then click on **Travel admin** from the top menu.
3. Continue to **Settings** > **Integrations**



4. From the **Navan API Credentials** section, select **Create New**.
5. A pop-up window will prompt you to enter a **Description**, which serves as the name for the credentials and will be reflected in the list of credentials for your company.
6. [Optional] Enter **Allowed IP Addresses** or **Subnet Mask**
7. Click **Create**
8. A new pop-up window will show you the API credentials. Click **Done** or select **Copy All** to copy the Description, Client ID, Secret Key, and Allowed IP Addresses to your clipboard.

***Note**: Once this pop-up is closed, the secret key will no longer be accessible. **API credential details are only accessible once**.*



Managing existing API credentials

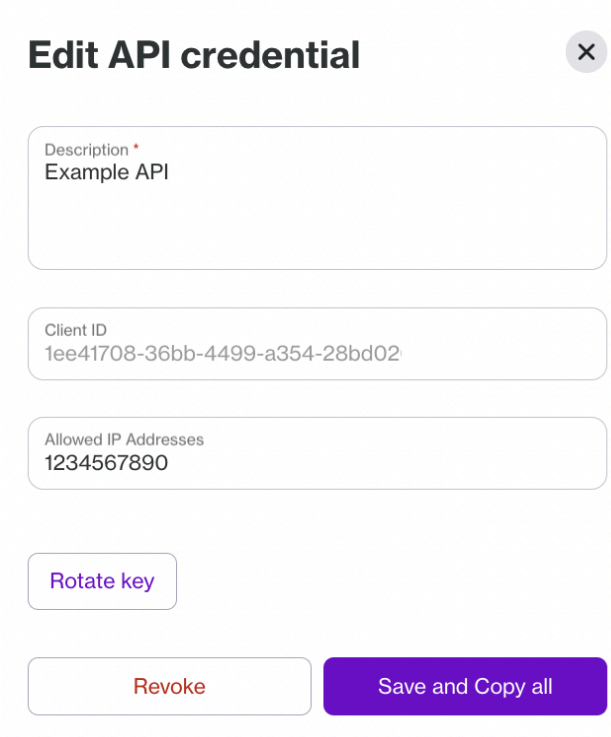
The Description and Client ID cannot be changed after creation; instead, the existing credential will need to be deleted, and a new one will need to be created.

Editing a credential

1. Select the **Edit** option associated with each credential
2. Edit the allowed IP addresses or subnet masks, rotate the key, or revoke the API credentials.
    A. An IP address or subnet mask may be replaced, but it is not possible to add an additional IP address or subnet mask.
    B. Selecting **Rotate Key** will change the private key for the same Client ID

Deleting a credential

1. Click **Edit** on an API credential and then select **Revoke**. You will be prompted to confirm that you would like to proceed with revoking the credential.
2. A popup will confirm that the credential was successfully deleted.



Keeping API keys safe

- Treat your secret API key as you would any other password
- Grant access only to those who need it; share and store this information via secure methods.
- Ensure it is kept out of any version control system you may be using
- Control access to your key using a password manager or secrets management service

Step 2: Obtain an access token

To obtain an access token, run the following command in Terminal.

Request

```
curl --request POST --url "https://api.navan.com/ta-auth/oauth/token" --header 'content-type:application/x-www-form-urlencoded' --data grant_type=client_credentials --data client_id={clientId} --data client_secret={clientSecret}
```

Note: For EU customers use url: https://app-fra.navan.com/ta-auth/oauth/token

Response

{"access_token":"{accessToken}","token_type":"bearer","expires_in":43199,"scope":"bookings:read liquid:read"}

This access token can be used to access data from both booking API and Expense API.

Step 3: Verify access token

To verify an access token, run the following command in Terminal.

Request

```
curl --request POST --url
"https://api.navan.com/ta-auth/oauth/check_token?token={accessToken}" -u
{clientId}:{clientSecret}

Note: For EU customers use url:
https://app-fra.navan.com/ta-auth/oauth/check_token?token={accessToken}
```

Response

{"scope":["bookings:read","liquid:read"],"active":true,"exp":1662834717,"client_id":"{clientId}"}

Step 4: Run Transaction API

To run Transaction API, run the following command in Terminal.

Request

```
curl -H "Authorization: Bearer {accessToken}"
"https://api.navan.com/v1/liquid/reports?date=2022-09-01&reportType=TRANSACTIONS" or
curl -H "Authorization: Bearer {accessToken}"
"https://api.navan.com/v1/liquid/reports?reportType=TRANSACTIONS&dateModified=2022-09-15" or
curl -H "Authorization: Bearer {accessToken}"
"https://api.navan.com/v1/liquid/reports?date=2022-09-01&reportType=TRANSACTIONS&dateModified=2022-09-15" or
curl -H "Authorization: Bearer {accessToken}"
"https://api.navan.com/v1/liquid/reports?reportType=TRANSACTIONS&lastApproverActionDate=2022-09-15&approvalStatus=APPROVED"

Note: For EU customers:
curl --location
'https://app-fra.navan.com/v1/liquid/reports?date=2023-12-10&reportType=TRANSACTIONS' \
--header 'Authorization: Bearer {accessToken}' \
--header 'X-ta-region: EU'
```

Parameters

`date` : **YYYY-MM-DD (Ex: 2022-09-01)**

`reportType` : below are the different report types or transaction Types

- **TRANSACTIONS** - (Navan Card transactions)
- **MANUAL_TRANSACTIONS** - (Manual Reimbursement Transactions)
- **REPAYMENTS -** (Repayment transactions)
- **CONNECT -** (Other card programs - Amex, citi etc)

`dateModified`: **YYYY-MM-DD (Ex: 2022-09-01) -** transaction modified date

lastApproverActionDate: YYYY-MM-DD (Ex: 2022-09-01)
approvalStatus: valid values - APPROVED,REJECTED,PARTIALLY_REJECTED and FLAGGED

Depending on the **reportType** parameter, **date** parameter logic changes.

reportType = "**MANUAL_TRANSACTIONS**"

- It fetches all **REIMBURSEMENT Initiated** transactions. **REIMBURSEMENT_DATE and POSTED_DATE are the same for manual Transactions. In case of PAYROLL transactions, POSTED_DATE is NULL and you only see REIMBURSEMENT_DATE.**
- **date:** parameter value will be **posted_date (for PAYROLL Transaction its REIMBUSEMENT_DATE)**
  Note: In API response **Json fields:**
  - **posted_date** and **reimbursement_date** will hold Deposit Date.
  - **Last_approver_action_date** will be **approver action date** (Auto or user approve/reject or flagged action)

reportType = "**TRANSACTIONS" or "REPAYMENTS" or "CONNECT"**
It fetches all posted credit card transactions (Activity_Type: Purchase, Refund and Repayment)
date: parameter value will be **posted_date** (which is credit card **charged_date**)

Note: either **date** or **dateModified** or **lastApproverActionDate** parameter has to be passed.

Response

Json format output (if the access token validates successfully and api access is enabled for the customer)

Example


```
[
  {
    "ID": "txn_1Ld8PTDIPUWPFlfYfJ7",
    "SOURCE_ID": "ipi_1Ld8PRDIPU7WbLPqczPn",
    "ACTIVITY_TYPE": "Purchase",
    "ACTIVITY_DESCRIPTION": null,
    "POSTED_DATE": "2022-09-01",
    "AUTHORIZATION_DATE": "2022-08-30",
    "MANUALLY_ADDED_DATE": null,
    "REQUESTED_AMOUNT": null,
    "POSTED_AMOUNT": 3.30,
    "POSTED_CURRENCY": "EUR",
    "FEE": null,
    "ORIGINAL_AMOUNT": 3.30,
    "ORIGINAL_CURRENCY": "EUR",
    "CARDHOLDER": "xxxx xxxx",
    "CARDHOLDER_EMPLOYEE_ID": "1234",
    "CARDHOLDER_EMAIL": "abc@xyz.com",
    "CARDHOLDER_GUEST": false,
    "CARD_DESCRIPTION": "PHYSICAL CORPORATE CARD VISA 1234",
    "CARD_NICKNAME": null,
    "CARD_OWNER": "abc@xyz.com",
    "APPROVER_TYPE": "auto",
    "APPROVED_BY_EMAIL": null,
    "POLICY": "Traveling: meals for myself",
    "POLICY_CATEGORY": "On-trip expenses",
    "MERCHANT_CATEGORY": "eating_places_restaurants",
    "MERCHANT_NAME": "Wrights Food Court",
    "MERCHANT_ADDRESS": "Dublin, IE",
    "BILLABLE_ENTITY": "ABC Ireland Limited",
    "BILLABLE_ENTITY_AMOUNT": null,
    "BILLABLE_ENTITY_CURRENCY": null,
    "LEGAL_ENTITY": "ABC Ireland Limited",
    "LEGAL_ENTITY_AMOUNT": null,
```

```
"LEGAL_ENTITY_CURRENCY": null,
"TRANSACTION_DESCRIPTION": null,
"GL_CODE_NUMBER": "62400",
"GL_CODE_NAME": "Meals",
"TAXABLE_BENEFIT": false,
"PERSONAL": false,
"PERSONAL_AMOUNT": null,
"FLAGGED": false,
"FLAG_STATUS": null,
"FLAG_REASONS": "",
"REIMBURSEMENT_METHOD": null,
"AMOUNT_REPAID": 0,
"AMOUNT_REJECTED": 0,
"ADMIN_NOTE": null,
"TRIP_UUID": "78d6af2a-dklsdfkjh-b4f0-2ad152f7d99f",
"TRIP_NAME": "QBR Q2",
"TRIP_PURPOSE": "Berlin QBRs Q2",
"INVITER_EMPLOYEE_ID": "14143",
"INVITER_EMAIL": "abc@xyz.com",
"INVITER_NAME": "Tony Stark",
"INVITER_DEPARTMENT": "Sales",
"INVITER_REGION": "201 - London",
"INVITER_COST_CENTER": "2010 - Sales - Corporate",
"INVITER_SUBSIDIARY": "Stark Ltd",
"HOTEL_NAME": null,
"BOOKING_START_DATE": null,
"BOOKING_END_DATE": null,
"FLIGHT_ORIGIN_AIRPORT": null,
"FLIGHT_DESTINATION_AIRPORT": null,
"FLIGHT_CABIN": null,
"FLIGHT_MILES": null,
"TRAIN_MILES": null,
"BOOKING_STATUS": null,
"RELATED_TRAVELERS": null,
"TRAVELERS_EMPLOYEE_ID": "6354",
"TRAVELERS_REGION": "226 - Dublin",
"TRAVELERS_DEPARTMENT": "Sales",
"TRAVELERS_COST_CENTER": "2010 - Sales - Corporate",
"TRAVELERS_SUBSIDIARY": "ABC Ireland Limited",
"TRANSACTION_REGION": "226 - Dublin",
"TRANSACTION_DEPARTMENT": "Sales",
"TRANSACTION_COST_CENTER": "2010 - Sales - Corporate",
"TRANSACTION_SUBSIDIARY": "ABC Ireland Limited",
"BOOKERS": null,
"NAVAN_BOOKING_IDS": null,
"ETICKETS": null,
"RECEIPT": null,
"RECEIPT_NAVAN": null,
"ERECEIPT": "https://xx.yyy.com/companies/406/users/49d/receipts/ipi_1Ld8Pn/e-receipt.pdf?...",
"ERECEIPT_NAVAN": "https://app.Navan.com/app/Expense/admin/receipts/open…",
"PARTICIPANTS": "bmcevoy@Navan.com",
"VAT_NUMBER": null,
"COUNTRY_OF_TRANSACTION": null,
"TAX_TOTAL": null,
"TAX_TYPE_1": null,
"NET_AMOUNT_1": null,
"TAX_AMOUNT_1": null,
"TAX_TYPE_2": null,
"NET_AMOUNT_2": null,
"TAX_AMOUNT_2": null,
"TAX_TYPE_3": null,
"NET_AMOUNT_3": null,
"TAX_AMOUNT_3": null,
"VENDOR": "Wrights Food Court",
"DISTANCE_UNIT": null,
"DISTANCE_AMOUNT": null,
"EXCHANGE_RATE": 1,
"REIMBURSEMENT_EXCHANGE_RATE": null,
"ITEM_ID": "ipi_1Ld8PRDIPU7B57PWbLPqczPn",
"ITEM_TITLE": null,
"INVOICE_DATE": null,
"INVOICE_NUMBER": null,
"NET_AMOUNT": 3.30,
"ORIGINAL_TXN_ID": null,
"TA_INVOICE": null,
"REIMBURSEMENT_DATE": "2022-09-01",
"REIMBURSEMENT_AMOUNT": 75.00,
"REIMBURSEMENT_CURRENCY": "USD",
```

```
    "REPAID_TRANSACTION_ID": null,
    "ADJUSTMENT_DESCRIPTION": null,
    "HOTEL_PAYMENT_TYPE": null,
    "POSTED_DATE_TIME": "2023-04-24T11:07:28",
    "MODIFIED_TIMESTAMP": "2023-04-24T11:07:34",
    "FX_FEE_AMOUNT": null,
    "DIRECT_REIMBURSEMENT_FEE_AMOUNT": null,
    "STATEMENT_ID": null,
    "VCF_TRANSACTION_REFERENCE_NUMBER": null,
    "EST_DOC_DATE": null,
    "EST_DOC_NUMBER": null,
    "EST_DOC_LINK": null,
    "REBATE_TYPE": null,
    "ACCRUED_REBATE_AMOUNT": null,
    "APPROVAL_STATUS": null,
    "LAST_APPROVER_ACTION_DATE": null,
    "NAVAN_BOOKING_UUID": null,
    "REPORTING_TRANSACTION_ID": null,
    "EXTERNAL_TAX_CODE": null,
    "EXTERNAL_TAX_ID": null,
    "REPAYMENT_STATUS": null,
    "PAYOUT_ID": null,
    "TRANSACTION_TYPE": "Navan Physical",
    "CARD_PROGRAM_NAME": null,
    "<CUSTOM FIELD NAME>": "..."
    "<CUSTOM FIELD VALUE>": "..."
    "<CUSTOM FIELD LABEL>": "..."
    ...
  },
  {...
  }
]
```


Customer with no access to transaction API:

Response will be a 403 Forbidden Access:


```
{"timestamp":"2022-10-18T20:37:31.607+00:00",
 "Status":403,
 "error":"Forbidden",
 "message":"Forbidden","path":"/v1/liquid/reports"}
```

Paginated API

Customers can use paginated API if the no. of transactions per day is more than 1000 to avoid API timeout issues (504 Error).

```
curl -H "Authorization: Bearer {accessToken}"
"https://api.navan.com/v1/liquid/paginatedReports?date=2023-05-03&reportType=TRANSACTIONS"
```

Note: For EU customers:

```
curl --location
'https://app-fra.navan.com/v1/liquid/paginatedReports?date=2023-12-10&reportType=TRANSACTIONS' \
--header 'Authorization: Bearer {accessToken}' \
--header 'X-ta-region: EU'
```

Above call's response(json) will have below footer details:


```
{
    "content": [
        {
            "ID": "txn_1Ms3dYCWJXpWyfzkFwyq5BVO",
            "SOURCE_ID": "ipi_1Ms3dWCWJXpWyfzkdLy2fUwW",
            "ACTIVITY_TYPE": "Purchase",
            "ACTIVITY_DESCRIPTION": null,
            "POSTED_DATE": "2023-04-01",
            "AUTHORIZATION_DATE": "2023-03-30",
            "MANUALLY_ADDED_DATE": null,
            "REQUESTED_AMOUNT": null,
            "POSTED_AMOUNT": 70.00,
            "POSTED_CURRENCY": "USD",
            "FEE": null,
            "ORIGINAL_AMOUNT": 70.00,
            "ORIGINAL_CURRENCY": "USD",
            ...
        },
    ],
    "pageable": {
        "sort": {
            "sorted": false,
            "empty": true,
            "unsorted": true
        },
        "pageNumber": 0,
        "pageSize": 100,
        "offset": 0,
        "paged": true,
        "unpaged": false
    },
    "last": false,
    "totalElements": 1793,
    "totalPages": 18,
    "size": 100,
    "number": 0,
    "sort": {
        "sorted": false,
        "empty": true,
        "unsorted": true
    },
    "first": true,
    "numberOfElements": 100,
    "empty": false
}
```

No	Key	Description
1	pageNumber	Current page number (its starts with 0)
2	pageSize	Max No. of transactions per page (this can be set in API parameters as &size=500)
3	offset	Record pointer
4	last	Boolean value representing whether it's a last page or not.
5	totalElements	Total No. of transactions
6	totalPages	Total No.of pages
7	size	Same as pageSize (Max No. of transactions per page)
8	number	Same as pageNumber (current page number)
9	first	Boolean value representing whether it’s a first page or not.
10	numberOfElements	No. of transactions fetched for this page.

Below URL to change the pageSize to 500:

https://api.navan.com/v1/liquid/paginatedReports?date=2023-05-03&reportType=TRANSACTIONS&size=500

Below URL to fetch the second page data:

https://api.navan.com/v1/liquid/paginatedReports?date=2023-05-03&reportType=TRANSACTIONS&page=1

Sample Python Code

Below is a Python code snippet(NavanApi.py) that generates a json output file.

Python

```python
import json, os, sys, requests

# auth reference : https://docs.python-requests.org/en/master/api/#authentication
class RestUtil:
    def __init__(self, client_id, client_secret) -> None:
        self.CLIENT_ID = client_id
        self.CLIENT_SECRET = client_secret

    def expense_api(self, report_type, date="", date_modified="", region="US"):
        token_url = (
            "https://app-fra.navan.com/ta-auth/oauth/token"
            if region == "EU"
            else "https://api.navan.com/ta-auth/oauth/token"
        )
        response = requests.post(
            token_url,
            data={"grant_type": "client_credentials"},
            auth=(self.CLIENT_ID, self.CLIENT_SECRET),
        )
        try:
            json_output = json.dumps(response.json(), indent=4)
            token_value = response.json()["access_token"]
            print(json_output)
        except Exception as err:
            token_value = response.json()["error"]
            return json_output
        try:
            payload = {
                "date": date,
                "reportType": report_type,
                "dateModified": date_modified,
            }
            report_date = date_modified if date == "" else date

            file_path = (
                os.path.dirname(__file__)
                + report_type
                + "_"
                + report_date
                + ".json"
            )
```

```
            if region == "EU":
                BASE_URL = "https://app-fra.navan.com/v1/liquid/reports"
                headers = {
                    "Authorization": "Bearer {}".format(token_value),
                    "X-ta-region": "EU",
                }
            else:
                BASE_URL = "https://api.navan.com/v1/liquid/reports"
                headers = {"Authorization": "Bearer {}".format(token_value)}

            # expense API call
            auth_response = requests.get(BASE_URL, headers=headers, params=payload)

            # create output json file
            with open(file_path, "w", encoding="utf-8") as jsonf:
                jsonf.write(json.dumps(auth_response.json(), indent=4))
            return "Output file generated: " + file_path

        except (IOError, NameError, TypeError, RuntimeError, SyntaxError) as e:
            return f"{e}"

        except Exception as err:
            return f"Unexpected {err=}, {type(err)=}"
```

Python

```python
import NavanApi

# Enter API Credentials generate from Navan UI
# Travel Admin-> Settings -> Integrations -> Create API Credentials

rest_call = NavanApi.RestUtil(
    client_id='xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
    client_secret='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
)

response = rest_call.expense_api(
    report_type="MANUAL_TRANSACTIONS",
    date="2024-01-24",
    region="US",
)
print(response)
```

```
Above code Response: Output file generated: MANUAL_TRANSACTIONS_2024-01-24.json
```

Data Definition

json key / CSV column	Description	Nullable?	Data Type
ID	Transaction Unique Identifier	No	UUID - VARCHAR(50)
SOURCE_ID	Source ID from Navan	No	VARCHAR(50)
ACTIVITY_TYPE	Type of transaction in Expense (Purchase/Refund/Repayment/Manual Transaction)	No	VARCHAR(50)
ACTIVITY_DESCRIPTION	Only populated for virtual card purchases on the Navan platform	Yes	VARCHAR(50)
POSTED_DATE	Date the transaction was posted • Credit Card transactions it is charged Date • Reimbursement Date for manual transactions ***Note**: This field is populated for all transactions except **PAYROLL Transactions** (Use **REIMBUSEMENT_DATE** for PAYROLL transactions)*	Yes	DATE (10)
AUTHORIZATION_DATE	Date the transaction was initiated	No	DATE (10)
MANUALLY_ADDED_DATE	Date the employee added this expense to Navan Expense	Yes	DATE (20)
REQUESTED_AMOUNT	Amount the employee requested when they submitted the transaction	Yes	DECIMAL(19,2)
POSTED_AMOUNT	Final amount of the transaction (may be different from Requested amount if admin did not approve full amount) *FX Fee / Direct Reimbursement Fee is not included*	No	DECIMAL(19,2)
POSTED_CURRENCY	Currency that Navan Expense statement is paid in	No	VARCHAR(3)
FEE	Only relevant for Repayments. This is the passthrough amount due because Navan acts as the merchant for the repayment	Yes	DECIMAL(19,2)
ORIGINAL_AMOUNT	Amount of the transaction in local currency of the transaction *FX Fee / Direct Reimbursement Fee is not included*	No	DECIMAL(19,2)
ORIGINAL_CURRENCY	Local currency of the transaction	No	VARCHAR(3)
CARDHOLDER	Name of employee who submitted the transaction	No	VARCHAR(100)
CARDHOLDER_EMPLOYEE_ID	Employee ID of the employee who submitted the transaction	Yes	VARCHAR(255)
CARDHOLDER_EMAIL	Email address of the employee who submitted the transaction	No	VARCHAR(255)
CARDHOLDER_GUEST	Only relevant for Expense virtual card transactions. Will say TRUE for virtual cards issued in a Guest's name and FALSE for all other virtual cards	No	BOOLEAN
CARD_DESCRIPTION	Only relevant for Purchase cards. This is the user-provided description of the card	Yes	VARCHAR(100)
CARD_NICKNAME	Only relevant for Purchase cards. This is the user-provided name of the card	Yes	VARCHAR(100)
CARD_OWNER	Only relevant for Purchase cards. This is the employee responsible for the card	Yes	VARCHAR(255)
APPROVER_TYPE	Type of approval, can be auto (within policy), manager (policy requires manager approval if they have managers), Admin (Admin approved expense because it was flagged or was overriding manager review)	No	VARCHAR(100)
APPROVED_BY_EMAIL	Email address of the approver	Yes	VARCHAR(255)
POLICY	Policy applicable to this transaction, provided by the user in the case of manual transactions	Yes	VARCHAR(100)
POLICY_CATEGORY	Broad bucket of categories	No	VARCHAR(100)
MERCHANT_CATEGORY	Visa merchant category code	Yes	VARCHAR(255)

MERCHANT_NAME	Name of the merchant	Yes	VARCHAR(255)
MERCHANT_ADDRESS	Address of merchant as provided by the merchant. May not be the actual physical address	Yes	VARCHAR(1000)
BILLABLE_ENTITY	Shows the Legal entity of the cardholder	No	VARCHAR(100)
BILLABLE_ENTITY_AMOUNT	Amount of the transaction in billable currency of the transaction	Yes	DECIMAL(19,2)
BILLABLE_ENTITY_CURRENCY	Billable currency of the transaction	Yes	VARCHAR(3)
LEGAL_ENTITY	The legal entity the user is assigned to	No	VARCHAR(100)
LEGAL_ENTITY_AMOUNT	Amount of the transaction in legal entity currency of the transaction	Yes	DECIMAL(19,2)
LEGAL_ENTITY_CURRENCY	Legal entity currency of the transaction	Yes	VARCHAR(3)
TRANSACTION_DESCRIPTION	User-provided description	Yes	VARCHAR(1000)
GL_CODE_NUMBER	GL code based on the GL upload or GL mapping defined in Policy	Yes	VARCHAR(15)
GL_CODE_NAME	GL name based on the GL upload or GL mapping defined in Policy	Yes	VARCHAR(500)
TAXABLE_BENEFIT	**TRUE** if this GL code is defined as a Taxable Benefit; **FALSE** if not	**No**	BOOLEAN
PERSONAL	**TRUE** if transaction has been identified as Personal by user or admin; **FALSE** if not	**No**	BOOLEAN
PERSONAL_AMOUNT	Amount of the transaction that has been identified as Personal and therefore not reimbursed *FX Fee / Direct Reimbursement Fee is not included*	Yes	DECIMAL(19,2)
FLAGGED	Shows **TRUE** if the transaction was ever flagged, even if the flag has been resolved; **FALSE** if never flagged	No	BOOLEAN
FLAG_STATUS	Shows **APPROVED** if admin has approved a previously flagged transaction; Blank if never flagged. • **PENDING_CARDHOLDER**: Used when the user needs to provide additional information for a transaction such as custom field, receipt, etc. • **PENDING_APPROVER**: Used when approver needs to review a transaction and decide to approve it or not. • **APPROVED**: Used when a transaction is approved by approver and also when a transaction is auto approved by system. • **REJECTED**: Used when a manual transaction is rejected by approver. • **REJECTED_PENDING_REFUND**: Used when the user needs to repay all or part of the transaction amount. • **REJECTED_REFUNDED**: Used when the user has repaid all or part of the transaction amount. • **AUTO_REJECTED_PENDING_REFUND**: Used when a user needs to repay a transaction amount that was auto rejected by policy. • **APPEALED**: Used when a user appeals to an auto rejected transaction and such transaction needs to be reviewed by admin.	Yes	VARCHAR(100)
FLAG_REASONS	Lists all reasons this transaction was ever flagged	Yes	VARCHAR(1000)
REIMBURSEMENT_METHOD	Shows ACH if the user had connected their bank account at the time the transaction was submitted; Shows Payroll if not	Yes	VARCHAR(100)
AMOUNT_REPAID	Only relevant for Expense card transactions; shows the amount the user repaid their company *FX Fee / Direct Reimbursement Fee is not included*	Yes	DECIMAL(19,2)
AMOUNT_REJECTED	Shows the amount of the transaction that was rejected	Yes	DECIMAL(19,2)

	(could be partial or full amount) *FX Fee / Direct Reimbursement Fee is not included*		
ADMIN_NOTE	Any note added by the admin	Yes	VARCHAR(5000)
TRIP_UUID	unique ID of a Trip	Yes	UUID - VARCHAR(50)
TRIP_NAME	Only for travel bookings made on Navan, will show the name of the Trip	Yes	VARCHAR(50)
TRIP_PURPOSE	Only for travel bookings made on Navan, will show the purpose of the Trip	Yes	VARCHAR(500)
INVITER_EMPLOYEE_ID	For guest bookings only, shows employee ID of the person who invited the guest	Yes	VARCHAR(50)
INVITER_EMAIL	For guest bookings only, shows email of the person who invited the guest	Yes	VARCHAR(255)
INVITER_NAME	For guest bookings only, shows name of the person who invited the guest	Yes	VARCHAR(255)
INVITER_DEPARTMENT	For guest bookings only, shows department of the person who invited the guest	Yes	VARCHAR(255)
INVITER_REGION	For guest bookings only, shows region of the person who invited the guest	Yes	VARCHAR(255)
INVITER_COST_CENTER	For guest bookings only, shows cost center of the person who invited the guest	Yes	VARCHAR(255)
INVITER_SUBSIDIARY	For guest bookings only, shows subsidiary of the person who invited the guest	Yes	VARCHAR(255)
HOTEL_NAME	For hotel bookings made on Navan, shows the name of the hotel	Yes	VARCHAR(255)
BOOKING_START_DATE	For travel bookings made on Navan, shows the date the booking starts	Yes	DATE (10)
BOOKING_END_DATE	For travel bookings made on Navan, shows the date the booking ends	Yes	DATE (10)
FLIGHT_ORIGIN_AIRPORT	For flight bookings made on Navan, shows the origin airport	Yes	VARCHAR(50)
FLIGHT_DESTINATION_AIRPORT	For flight bookings made on Navan, shows the destination airport	Yes	VARCHAR(50)
FLIGHT_CABIN	For flight bookings made on Navan, shows the cabin booked	Yes	VARCHAR(255)
FLIGHT_MILES	For flight bookings made on Navan, shows the flight distance in miles	Yes	DOUBLE
TRAIN_MILES	For train bookings made on Navan, shows the trains distance in miles	Yes	DOUBLE
BOOKING_STATUS	For travel bookings made on Navan, shows the current status (PENDING, TICKETED, CANCELLED, VOIDED)	Yes	VARCHAR(50)
RELATED_TRAVELERS	For hotel bookings made on Navan with multiple employees per room, shows name of the other employee(s)	Yes	VARCHAR(5000)
TRAVELERS_EMPLOYEE_ID	Employee ID of the user who submitted the expense (not necessarily related to travel)	No	VARCHAR(50)
TRAVELERS_REGION	Region of the user who submitted the expense (not necessarily related to travel)	Yes	VARCHAR(50)
TRAVELERS_DEPARTMENT	Department of the user who submitted the expense (not necessarily related to travel)	Yes	VARCHAR(255)
TRAVELERS_COST_CENTER	Cost center of the user who submitted the expense (not necessarily related to travel)	Yes	VARCHAR(255)
TRAVELERS_SUBSIDIARY	Subsidiary of the user who submitted the expense (not necessarily related to travel)	Yes	VARCHAR(255)
TRANSACTION_REGION	Region on transaction for Purchase transactions or Non-Purchase transactions where the admin changed	Yes	VARCHAR(50)

	region from the default		
TRANSACTION_DEPARTMENT	Department on transaction for Purchase transactions or Non-Purchase transactions where the admin changed department from the default	Yes	VARCHAR(255)
TRANSACTION_COST_CENTER	Cost Center on transaction for Purchase transactions or Non-Purchase transactions where the admin changed cost center from the default	Yes	VARCHAR(255)
TRANSACTION_SUBSIDIARY	Subsidiary on transaction for Purchase transactions or Non-Purchase transactions where the admin changed subsidiary from the default	Yes	VARCHAR(255)
BOOKERS	For travel bookings made on Navan by someone other than the employee (e.g. an EA or delegate), will show the name of the person who made the booking	Yes	VARCHAR(2048)
NAVAN_BOOKING_IDS	For travel bookings made on Navan, will show the Navan booking ID	Yes	VARCHAR(2048)
ETICKETS	For flight bookings made on Navan, will show the e-ticket number	Yes	VARCHAR(2048)
RECEIPT	Link to a downloadable receipt hosted on Amazon AWS. Link expires 7 days after export. If syncing to ERP, make sure the destination field can accept up to 512 characters.	Yes	VARCHAR(2048)
RECEIPT_NAVAN	Link to a receipt hosted in Navan Expense and viewable in the browser. Requires Expense Admin role to view. Link does not expire. If syncing to ERP, make sure the destination field can accept up to 512 characters.	Yes	VARCHAR(2048)
ERECEIPT	Link to a downloadable e-receipt hosted on Amazon AWS. Link expires 7 days after export. If syncing to ERP, make sure the destination field can accept up to 512 characters.	Yes	VARCHAR(2048)
ERECEIPT_NAVAN	Link to an e-receipt hosted in Navan Expense and viewable in the browser. Requires Expense Admin role to view. Link does not expire. If syncing to ERP, make sure the destination field can accept up to 2048 characters.	Yes	VARCHAR(2048)
PARTICIPANTS	All users listed on a transaction	Yes	VARCHAR(2048)
VAT_NUMBER	tax id of the merchant	Yes	VARCHAR(50)
COUNTRY_OF_TRANSACTION	highlights the country of the merchant the txn occurred in	No	VARCHAR(255)
TAX_TOTAL	Total VAT/GST	Yes	DECIMAL(19,2)
TAX_TYPE_1	Name of first tax line(VAT 10%/ VAT 20%/GST/etc.)	Yes	VARCHAR(50)
NET_AMOUNT_1	Net amount of first tax line	Yes	DECIMAL(19,2)
TAX_AMOUNT_1	Tax Amount of first tax line	Yes	DECIMAL(19,2)
TAX_TYPE_2	Name of second tax line(VAT 10%/ VAT 20%/GST/etc.)	Yes	VARCHAR(50)
NET_AMOUNT_2	Net amount of second tax line	Yes	DECIMAL(19,2)
TAX_AMOUNT_2	Tax Amount of second tax line	Yes	DECIMAL(19,2)
TAX_TYPE_3	Name of first tax (VAT 10%/ VAT 20%/GST/etc.)	Yes	VARCHAR(50)
NET_AMOUNT_3	Net amount of 3rd tax line	Yes	DECIMAL(19,2)
TAX_AMOUNT_3	Tax Amount of 3rd tax line	Yes	DECIMAL(19,2)
VENDOR	Cleaned up version of the original merchant_name	No	VARCHAR(255)
DISTANCE_UNIT	Distance Unit	Yes	VARCHAR(50)
DISTANCE_AMOUNT	DIstance Amount *FX Fee / Direct Reimbursement Fee is not included*	Yes	DECIMAL(19,2)
EXCHANGE_RATE	Exchange Rate used on the transaction.	No	DOUBLE
REIMBURSEMENT_EXCHANGE_RATE	Exchange Rate between Original Currency and employees Reimbursed Currency	Yes	DOUBLE

ITEM_ID	Itemization (unique Key)	No	VARCHAR(50)
ITEM_TITLE	Itemization description	Yes	VARCHAR(2048)
INVOICE_DATE	Date of Invoice	Yes	DATE (10)
INVOICE_NUMBER	Invoice Number	Yes	VARCHAR(50)
NET_AMOUNT	Net Amount *FX Fee / Direct Reimbursement Fee is not included*	No	DECIMAL(19,2)
ORIGINAL_TXN_ID	Transaction ID for the refund Transaction	Yes	UUID - VARCHAR(50)
TA_INVOICE	Navan Travel Invoice URL	Yes	VARCHAR(5000)
REIMBURSEMENT_DATE	Reimbursement Date ((only for manual Transactions)	Yes	DATE (10)
REIMBURSEMENT_AMOUNT	Reimbursement Amount ((only for manual Transactions) *FX Fee / Direct Reimbursement Fee is not included*	Yes	DECIMAL(19,2)
REIMBURSEMENT_CURRENCY	Reimbursement Currency (only for manual Transactions)	Yes	VARCHAR(3)
REPAID_TRANSACTION_ID	Transaction ID of the repaid amount by the employee	Yes	UUID - VARCHAR(50)
ADJUSTMENT_DESCRIPTION	Adjustment Description	Yes	VARCHAR(2048)
HOTEL_PAYMENT_TYPE	Hotel payment type (Pay Later or Pay Now)	Yes	VARCHAR(50)
POSTED_DATE_TIME	Date and time the transaction was posted ***Note**: This field is populated for all transactions except for **PAYROLL Transactions***	Yes	DATETIME (20)
MODIFIED_TIMESTAMP	Date and time transaction data was modified.	No	DATETIME (20)
FX_FEE_AMOUNT	Foreign Exchange (FX) Fees charge for EUR/GBP cards (only for card transactions - Purchase activity type)	Yes	DECIMAL(19,2)
DIRECT_REIMBURSEMENT_FEE_AMOUNT	Direct Reimbursement Fee amount (only for manual transactions)	Yes	DECIMAL(19,2)
STATEMENT_ID	Statement ID	Yes	VARCHAR(50)
VCF_TRANSACTION_REFERENCE_NUMBER	Transaction Reference number (only for CONNECT transactions)	Yes	VARCHAR(50)
EST_DOC_DATE	The latest date on which the estimated charges document is generated or updated. Format YYYY-MM-DD	Yes	DATE(10)
EST_DOC_NUMBER	A unique identifier for every version of the estimated charges document.	Yes	VARCHAR(250)
EST_DOC_LINK	Link to estimated charges document	Yes	VARCHAR(4000)
REBATE_TYPE	Rebate type (HOTEL or OTHER)	Yes	VARCHAR(50)
ACCRUED_REBATE_AMOUNT	Accrued Rebate Amount	Yes	DECIMAL(19,2)
APPROVAL_STATUS	The transaction's approval state. Transactions can have an approval status of Approved, Flagged, In review, or Rejected. • **APPROVED**: The physical, virtual card or manually submitted transaction has been approved and is ready to be exported or synced • **FLAGGED**: Flagged transactions may violate the company's set expense policies and require additional review or action • **IN_REVIEW**: The transaction falls outside of the parameters set by the assigned expense policy and is being reviewed by the approver • **REJECTED**: The transaction falls outside of the parameters set by the assigned expense policy and has been rejected after review by the approver. If the transaction was submitted as a manual (out-of-pocket) expense, it will not be reimbursed. If the transaction was charged to a Navan card, the employee will be prompted to	No	VARCHAR(100)

	reimburse the company through the Navan app		
LAST_APPROVER_ACTION_DATE	The date the transaction was auto-approved or when the approver last took action—**approved, rejected, or requested repayment**. **Note**: If more information is requested, this date is not updated.	No	DATETIME(20)
NAVAN_BOOKING_UUID	A unique identifier (UUID) generated internally by Navan for each travel booking	Yes	VARCHAR(50)
REPORT_TRANSACTION_ID	Serves as the external transaction ID. Specifically for Connect feeds such as VCF, CDF, or Amex GL.	Yes	VARCHAR(50)
EXTERNAL_TAX_CODE	Tax code reference uploaded by Admin.	Yes	VARCHAR(100)
EXTERNAL_TAX_ID	Tax ID reference uploaded by Admin.	Yes	VARCHAR(100)
REPAYMENT_STATUS	Current status of each repayment transaction	Yes	VARCHAR(35)
PAYOUT_ID	A unique reference number assigned to each payout transaction or batch. This ID can be used to reconcile repayment transactions with the corresponding deposits in the company's bank account.	Yes	VARCHAR(50)
TRANSACTION_TYPE	Identifies if the transaction type is: Manual, Connect Physical, Connect Travel Payments, Navan Physical, Navan Travel Payments, or Navan Purchase Card.	No	VARCHAR(50)
CARD_PROGRAM_NAME	Shows the name of the credit card program.	Yes	VARCHAR(150)
CUSTOM FIELD NAME		Yes	VARCHAR(2000)
CUSTOM FIELD VALUE		Yes	VARCHAR(2000)
CUSTOM FIELD LABEL		Yes	VARCHAR(2000)