Downloading reports

Our reporting platform has moved to new infrastructure, and the way automated scripts sign in has changed. This guide shows how to sign in, download a report as JSON, and optionally convert it to CSV. It ends with a complete script you can schedule.

What you need

- A Linux or macOS machine. On Windows, see Help for Windows users.
- **cURL**, which sends the web requests, and **jq**, which reads and reshapes the JSON results.
- The user name and password you use to sign in to the reporting portal. Users who sign in only through single sign-on (SSO) can't use this method. Ask your account representative for a dedicated integration user if you need one.
- Your portal address. Sign in to the portal in a browser and look at the address bar. Copy everything up to and including the first part of the path, which identifies your company. If the address bar shows `https://reports.example.com/yourcompany/…`, your portal address is `https://reports.example.com/yourcompany`. Use exactly the host your browser shows.

The examples use these shell variables. Set `BASE_URL` to your portal address:

```
BASE_URL="https://reports.example.com/yourcompany"
CRED_DIR="$HOME/.report-api"
```

Step by step

1. Create a private folder and a credentials file

The folder holds your password and your session cookie, so only your user account may read it.

```
mkdir -p "$CRED_DIR"
chmod 700 "$CRED_DIR"
(umask 077; touch "$CRED_DIR/credentials.json")
nano "$CRED_DIR/credentials.json"
```

In the editor, enter one line with your user name and password, then save:

```
{"user_name": "jsmith@example.com", "password": "your-password"}
```

Check that only you can read it. The listing should start with `-rw-------`:

```
ls -l "$CRED_DIR/credentials.json"
```

Use an editor rather than `echo '{…}' > file`. A password typed into a command is saved in your shell history.

2. Sign in

cURL sends the credentials file and saves the session cookie the portal returns. It reads the password from the file, so the password never appears on the command line, where other users on the machine could see it.

```
curl -sS -o /dev/null -w "HTTP %{http_code}\n" \
  -c "$CRED_DIR/cookies.txt" \
  -H "Content-Type: application/json" \
  -d @"$CRED_DIR/credentials.json" \
  "$BASE_URL/api/login"
```

HTTP 204

Signed in. The session cookie is saved in `cookies.txt`.

HTTP 400

The user name or password is wrong, or the credentials file is not valid JSON. Also check that `BASE_URL` matches the address your browser shows, including your company's path.

3. Download a report as JSON

This saves the full response, then keeps only its `d` field, which holds the report. `--fail` makes cURL stop with an error if the request is refused, instead of saving an error page as your report.

```
curl -sS --fail -b "$CRED_DIR/cookies.txt" \
  -o admin_charge.raw.json \
  "$BASE_URL/api/report/AdminChargeDetail" \
  && jq '.d' admin_charge.raw.json > admin_charge.json
```

The result is a JSON list with one object per report row. Each key is a column's data name. The report's schema (see Finding reports and their columns) maps each data name to the heading the portal shows.

This report covers the most recent statement month unless you ask for another. To choose the month, add `statement_month` set to the first day of that month:

```
curl -sS --fail -b "$CRED_DIR/cookies.txt" \
  -o admin_charge.raw.json \
  --get --data-urlencode "statement_month=2026-08-01" \
  "$BASE_URL/api/report/AdminChargeDetail"
```

Other reports may take different parameters. Ask your account representative which ones a report supports.

4. Optionally, convert the report to CSV

The first line is the column headers, in the order the report returns them. Values containing commas or quotes are escaped correctly.

```
jq -r '
  def cell: if type == "object" or type == "array"
            then tojson else . end;
  (.[0] // {} | keys_unsorted) as $cols
  | $cols, (.[] | [ .[$cols[]] | cell ])
  | @csv
' admin_charge.json > admin_charge.csv
```

To use the portal's column headings instead, combine the report with its schema. This keeps only the columns marked for export, in the schema's order, headed by their portal titles:

```
curl -sS --fail -b "$CRED_DIR/cookies.txt" \
  -o admin_charge.schema.json \
  "$BASE_URL/api/report_schema/AdminChargeDetail" \
  && jq -r --slurpfile s admin_charge.schema.json '
  def cell: if type == "object" or type == "array"
            then tojson else . end;
  [ $s[0].d.columns[] | select(.show_export != false) ] as $cols
  | ($cols | map(.title)),
    (.[] | [ .[$cols[].dataset] | cell ])
  | @csv
' admin_charge.json > admin_charge.csv
```

5. Sign out and remove the cookie

Do this when you are done.

```
curl -sS -o /dev/null -b "$CRED_DIR/cookies.txt" "$BASE_URL/api/logout"
rm -f "$CRED_DIR/cookies.txt" \
  admin_charge.raw.json admin_charge.schema.json
```

Finding reports and their columns

Run these while you are signed in (after step 2).

List the reports you can use. The result lists each report's name, which is what goes in the URL, along with its display name and category. This list is available to administrator and manager users. If it comes back empty, ask your account representative.

```
curl -sS --fail -b "$CRED_DIR/cookies.txt" \
  "$BASE_URL/api/reportslist" | jq '.d'
```

Look up a report's schema. The schema describes a report's columns without running it. Your company's own names for cost centers and similar fields appear in the headings.

```
curl -sS --fail -b "$CRED_DIR/cookies.txt" \
  "$BASE_URL/api/report_schema/AdminChargeDetail" \
  | jq '.d.columns[] | {dataset, title, type, show_export}'
```

Field	Meaning
`dataset`	The column's data name. This is the key used in each report row.
`title`	The column heading the portal shows.
`type`	The kind of value, such as `string`.
`show_export`	Whether the portal includes the column when you export the report.

The columns are listed in the report's defined order.

Complete script for scheduled downloads

This script does steps 2 to 5 in one run. It stops with an error message if sign-in or the download fails, and it always signs out and removes the cookie. Create the credentials file first (step 1), then set `BASE_URL`, `REPORT` and `OUT_DIR` at the top.

get-report.sh

```
#!/usr/bin/env bash
# Download one report as JSON and CSV.
set -euo pipefail
umask 077

BASE_URL="https://reports.example.com/yourcompany"
CRED_DIR="$HOME/.report-api"
REPORT="AdminChargeDetail"
OUT_DIR="$HOME/reports"

COOKIES="$CRED_DIR/cookies.txt"
RAW="$OUT_DIR/$REPORT.raw.json"
```

```
mkdir -p "$OUT_DIR"

cleanup() {
  curl -sS -o /dev/null -b "$COOKIES" "$BASE_URL/api/logout" || true
  rm -f "$COOKIES" "$RAW"
}
trap cleanup EXIT

echo "Signing in to $BASE_URL"
code=$(curl -sS -o /dev/null -w '%{http_code}' -c "$COOKIES" \
  -H "Content-Type: application/json" \
  -d @"$CRED_DIR/credentials.json" \
  "$BASE_URL/api/login")
if [ "$code" != "204" ]; then
  echo "Sign-in failed (HTTP $code)" >&2
  exit 1
fi

echo "Downloading the $REPORT report. A large report can take several"
echo "minutes, and nothing more is shown until it finishes."
curl -sS --fail -b "$COOKIES" -o "$RAW" "$BASE_URL/api/report/$REPORT"

jq '.d' "$RAW" > "$OUT_DIR/$REPORT.json"

jq -r '
  def cell: if type == "object" or type == "array"
            then tojson else . end;
  (.[0] // {} | keys_unsorted) as $cols
  | $cols, (.[] | [ .[$cols[]] | cell ])
  | @csv
' "$OUT_DIR/$REPORT.json" > "$OUT_DIR/$REPORT.csv"

echo "Saved $OUT_DIR/$REPORT.json and $OUT_DIR/$REPORT.csv"
```

> This script is also attached to this PDF as `get-report.sh`. In Adobe Acrobat or Apple Preview, open the attachments panel to save it directly instead of copying it.

Save it as `get-report.sh`, run `chmod 700 get-report.sh`, and schedule it with cron or your usual job scheduler.

The script prints a line as it signs in and another as it starts the download. A large report can take several minutes to prepare, and nothing more appears while it does. That's normal, not a failure. The run is finished when the script prints the two files it saved.

Help for Windows users

Windows already includes what you need. The simplest route is the PowerShell script below. It needs nothing installed, keeps the session in memory instead of in a cookie file, and stores your password encrypted for your Windows account. It works in Windows PowerShell 5.1, which comes with Windows 10 and 11, and in PowerShell 7.

1. Save your credentials

Open PowerShell and run the following. Enter your portal user name and password when asked.

```
$dir = Join-Path $env:USERPROFILE '.report-api'
New-Item -ItemType Directory -Force -Path $dir | Out-Null
Get-Credential -Message 'Reporting portal sign-in' |
    Export-Clixml -Path (Join-Path $dir 'credential.xml')
```

Windows encrypts the password so that only your Windows account, on this computer, can read it. Run this while signed in to Windows as the account that will run the script. If you schedule the script to run as another account, sign in as that account to save the credentials. Run it again whenever your portal password changes.

2. Save the script

Paste it into Notepad and save it as `Get-Report.ps1`, for example in `C:\Users\<you>\report-api`. Then change the settings at the top:

- `$BaseUrl`: your portal address.
- `$Report`: the report to download.
- `$OutDir`: the folder the files are saved in. By default this is a `Reports` folder inside your Documents folder, created on the first run. To save somewhere else, replace the whole line with the folder's full path in single quotes, for example `$OutDir = 'D:\Finance\Reports'`.

Get-Report.ps1

```
# Download one report as JSON and CSV.
# Works in Windows PowerShell 5.1 and PowerShell 7.
$ErrorActionPreference = 'Stop'

# Your portal address and the report to download.
$BaseUrl = 'https://reports.example.com/yourcompany'
$Report  = 'AdminChargeDetail'

# The folder the JSON and CSV files are saved in. The default is a Reports
# folder in your Documents folder. To use another folder, replace the line
# with its full path in single quotes, for example:
#   $OutDir = 'D:\Finance\Reports'
$OutDir = Join-Path ([Environment]::GetFolderPath('MyDocuments')) 'Reports'

# Where step 1 saved your credentials. Leave this as it is.
$CredDir = Join-Path $env:USERPROFILE '.report-api'

# Older Windows PowerShell setups may not offer TLS 1.2 unless asked.
[Net.ServicePointManager]::SecurityProtocol =
    [Net.ServicePointManager]::SecurityProtocol -bor [Net.SecurityProtocolType]::Tls12

# Windows PowerShell 5.1 slows large downloads badly while it draws a
# progress bar. The script prints its own messages instead.
$ProgressPreference = 'SilentlyContinue'
```

```
$Invariant = [Globalization.CultureInfo]::InvariantCulture
$Utf8NoBom = New-Object System.Text.UTF8Encoding $false
$Utf8Bom   = New-Object System.Text.UTF8Encoding $true

# One CSV cell: nested values as JSON text, dates as ISO text,
# numbers with a "." decimal point whatever the machine's language.
function Format-Cell($Value) {
    if ($null -eq $Value) { return '' }
    if ($Value -is [string]) { return $Value }
    if ($Value -is [bool]) { return $Value.ToString().ToLowerInvariant() }
    if ($Value -is [datetime]) {
        return $Value.ToString('yyyy-MM-ddTHH:mm:ss.FFFFFFF', $Invariant)
    }
    if ($Value -is [System.Management.Automation.PSCustomObject] -or
        $Value -is [System.Collections.IEnumerable]) {
        return (ConvertTo-Json -InputObject $Value -Compress -Depth 20)
    }
    if ($Value -is [IFormattable]) { return $Value.ToString($null, $Invariant) }
    return "$Value"
}

$credential = Import-Clixml (Join-Path $CredDir 'credential.xml')
New-Item -ItemType Directory -Force -Path $OutDir | Out-Null
$raw = Join-Path $OutDir "$Report.raw.json"
$session = $null

try {
    $body = @{
        user_name = $credential.UserName
        password  = $credential.GetNetworkCredential().Password
    } | ConvertTo-Json -Compress
    Write-Host "Signing in to $BaseUrl"
    try {
        Invoke-RestMethod -Method Post -Uri "$BaseUrl/api/login" -UseBasicParsing `
            -ContentType 'application/json' -Body $body `
            -SessionVariable session | Out-Null
    } catch {
        throw "Sign-in failed: $($_.Exception.Message)"
    } finally {
        Remove-Variable body -ErrorAction SilentlyContinue
    }

    # Save the response as received, then read it as UTF-8. The server does
    # not name a character set, and Windows PowerShell 5.1 would otherwise
    # misread accented characters.
    Write-Host "Downloading the $Report report. A large report can take several"
    Write-Host "minutes, and nothing more is shown until it finishes."
    Invoke-RestMethod -Uri "$BaseUrl/api/report/$Report" -UseBasicParsing `
        -WebSession $session -OutFile $raw
    $rows = @((Get-Content -Path $raw -Raw -Encoding UTF8 | ConvertFrom-Json).d)

    $json = Join-Path $OutDir "$Report.json"
    $jsonText = ConvertTo-Json -InputObject $rows -Depth 20
    [IO.File]::WriteAllText($json, $jsonText, $Utf8NoBom)

    # The CSV starts with a UTF-8 byte-order mark so Excel reads accented
    # characters correctly.
    $csv = Join-Path $OutDir "$Report.csv"
    $csvText = ''
    if ($rows.Count -gt 0) {
        $columns = @($rows[0].PSObject.Properties.Name)
        $lines = $rows | ForEach-Object {
```

```
            $row = $_
            $cells = [ordered]@{}
            foreach ($column in $columns) { $cells[$column] = Format-Cell $row.$column }
            [pscustomobject]$cells
        } | ConvertTo-Csv -NoTypeInformation
        $csvText = ($lines -join "`r`n") + "`r`n"
    }
    [IO.File]::WriteAllText($csv, $csvText, $Utf8Bom)

    Write-Output "Saved $json and $csv"
} finally {
    if ($session) {
        try {
            Invoke-RestMethod -Uri "$BaseUrl/api/logout" -UseBasicParsing `
                -WebSession $session | Out-Null
        } catch { }
    }
    Remove-Item -Path $raw -ErrorAction SilentlyContinue
}
```

> This script is also attached to this PDF as `Get-Report.ps1`. In Adobe Acrobat, open the attachments panel to save it directly instead of copying it.

The CSV opens directly in Excel with accented characters intact. Dates are written as `2026-08-01` or `2026-08-01T12:34:56`, and decimals always use a period, whatever language Windows is set to.

3. Run it

In PowerShell, go to the folder that holds the script and run:

```
powershell.exe -NoProfile -ExecutionPolicy RemoteSigned -File .\Get-Report.ps1
```

The script prints a line as it signs in and another as it starts the download. A large report can take several minutes to prepare, and nothing more appears while it does. That's normal, not a failure. The run is finished when the script prints the two files it saved.

`-ExecutionPolicy RemoteSigned` lets scripts you created on this computer run, for this run only, without changing any system setting. If your organization's IT policy blocks scripts, ask your IT team. If you downloaded the script instead of pasting it, Windows marks it as coming from the internet. Run `Unblock-File .\Get-Report.ps1` once to allow it.

4. Schedule it

Use Task Scheduler:

- Choose **Create Task**. On the General tab, choose the account that saved the credentials and select **Run whether user is logged on or not**. Leave **Do not store password** unticked; without the stored password, the script can't decrypt the saved credentials.
- On the Triggers tab, add the schedule you want.

- On the Actions tab, choose **Start a program**. Set the program to `powershell.exe` and the arguments to:

```
-NoProfile -ExecutionPolicy RemoteSigned -File "C:\Users\<you>\report-api\Get-Report.ps1"
```

Using cURL and jq on Windows

The cURL and jq steps earlier in this guide also work on Windows 10 and 11, which include cURL. Watch for these differences:

- **Type `curl.exe`, not `curl`.** In Windows PowerShell, `curl` runs a different built-in command that rejects cURL's options.
- **Install jq** with `winget install jqlang.jq`.
- **Put jq filters in a file**, such as `to-csv.jq`, and run `jq -r -f to-csv.jq report.json`. Command Prompt doesn't accept single quotes around a filter, and Windows PowerShell 5.1 can strip the double quotes inside one.
- **Write each command on one line**, or end continued lines with a backtick (`` ` ``) in PowerShell or a caret (`^`) in Command Prompt instead of `\`.
- **Replace Unix names:** `$HOME` becomes `$env:USERPROFILE` in PowerShell or `%USERPROFILE%` in Command Prompt, and `/dev/null` becomes `NUL`. Your user profile folder is private to your account by default, so `chmod` and `umask` aren't needed there.
- **Save jq output from Command Prompt.** In Windows PowerShell 5.1, `>` saves files as UTF-16 and can damage accented characters. Command Prompt saves them unchanged.

If something goes wrong

Result	What it means
`400` at sign-in	Wrong user name or password, a malformed credentials file, or the wrong company path in `BASE_URL`.
`403` on a report	The session has ended or was never started. Sign in again and retry once. If it still fails, this user doesn't have access to that report.
`404` on a report	No report with that name. Check the spelling and capitalization.
Empty CSV	The report returned no rows for your filters.

Keeping your access safe

- **Treat the cookie like the password.** Anyone who has the cookie can act as that user until it is signed out. Keep it in the private folder and delete it after each run.
- **Keep the credentials file private.** Only your user account should be able to read it (`-rw-------`). Store it outside shared folders such as `/tmp`, and leave it out of backups that other people can read.
- **Sign in once per run** and reuse the cookie for every download in that run. Heavy bursts of requests from one IP address are blocked automatically.
- **Send a User-Agent header.** cURL sends one automatically. If you port this to another HTTP tool, make sure it does too, or requests are rejected.

Questions or trouble signing in: contact your account representative.