Introduction

Resources

NFT API

Transfers API (Tx History)

Transaction Receipts API

Token API

Subgraphs

Webhooks

Websockets

Trace API

Debug API

ACCOUNT ABSTRACTION

- AA-SDK
- Account Abstraction FAQ

Embedded Accounts

- Accounts API Endpoints
-
  - Create Account post
-
  - Send Auth Email post
-
  - Authenticate User post
-
  - Get User post
-
  - Sign Message post
-
  - Register New Authenticator post

Gas Manager Admin API

- Gas Manager Admin API Quickstart
- Gas Manager Admin API Endpoints
-
  - Create Policy post
-
  - Get Policy get
-
  - Delete Policy delete
-
  - Replace Policy put
-
  - Get All Policies get
-
  - Update Policy Status put
-
  - Get Policy Stats get
-
  - Get Sponsorships get

Alchemy Transact

- Transact Quickstart
- Reinforced Transactions
- Transaction Simulation
-
  - Asset Changes
-
  - Execution Simulation
-
  - Bundle Simulation
-
  - Transaction Simulation Examples
-
  - Transaction Simulation FAQs
- Transaction Simulation Endpoints
-
  - alchemy_simulateAssetChanges post
-
  - alchemy_simulateAssetChangesBundle post
-
  - alchemy_simulateExecution post
-
  - alchemy_simulateExecutionBundle post
- Gas Optimized Transactions
-
  - alchemy_getGasOptimizedTransactionStatus post

Alchemy SDK

  - getTokenMetadata - SDK
-
  - getTransactionReceipt - SDK
-
  - send - SDK
-
  - getTransactionReceipts - SDK
-
  - getTransaction - SDK
-
  - isContractAddress - SDK
-
  - getNftMetadata - SDK
-
  - getNftSales - SDK
-
  - cancelPrivateTransaction - SDK
-
  - sendPrivateTransaction - SDK
-
  - traceTransaction - SDK
-
  - simulateExecutionBundle - SDK
-
  - simulateExecution - SDK
-
  - getMaxPriorityFeePerGas - SDK
-
  - simulateAssetChangesBundle - SDK
-
  - estimateGas - SDK
-
  - simulateAssetChanges - SDK
-
  - traceBlock - SDK
-
  - waitForTransaction - SDK
-
  - traceCall - SDK
-
  - sendTransaction - SDK
-
  - updateWebhook - SDK
-
  - refreshNftMetadata -SDK
-
  - createWebhook - SDK
-
  - getNftFilters - SDK
-
  - getAddresses - SDK
-
  - summarizeNftAttributes - SDK
-
  - deleteWebhook - SDK
-
  - searchContractMetadata - SDK
-
  - getAllWebhooks - SDK
-
  - getNftsForOwnerIterator - SDK
-
  - getNftsForContractIterator -SDK
-
  - getContractMetadata - SDK
-
  - getTransfersForOwner - SDK
-

Ethereum

  - net_version - Ethereum post
-
  - web3_clientVersion - Ethereum post
-
  - web3_sha3 - Ethereum post
-
  - eth_getTransactionByBlockNumberAndIndex - Ethereum post
-
  - eth_call - Ethereum post
-
  - eth_getBlockByHash - Ethereum post
-
  - eth_createAccessList - Ethereum post
-
  - eth_newFilter - Ethereum post
-
  - eth_getFilterChanges - Ethereum post
-
  - eth_getFilterLogs - Ethereum post
-
  - eth_newBlockFilter - Ethereum post
-
  - eth_newPendingTransactionFilter - Ethereum post
-
  - eth_uninstallFilter - Ethereum post
-
  - eth_subscribe
-
  - eth_unsubscribe

Polygon PoS

- Polygon PoS API Quickstart
-
  - Polygon SDK Examples
- Polygon PoS API FAQ
- Polygon PoS API Endpoints
-
  - bor_getAuthor - Polygon PoS post
-
  - bor_getCurrentProposer - Polygon PoS post
-
  - bor_getCurrentValidators - Polygon PoS post
-
  - bor_getRootHash - Polygon PoS post
-
  - eth_accounts - Polygon PoS post
-
  - eth_call - Polygon PoS post
-
  - eth_chainId - Polygon PoS post
-
  - eth_estimateGas - Polygon PoS post
-
  - eth_gasPrice - Polygon PoS post
-
  - eth_getBalance - Polygon PoS post
-
  - eth_getBlockByHash - Polygon PoS post
-
  - eth_getBlockByNumber - Polygon PoS post
-
  - eth_getBlockTransactionCountByHash - Polygon PoS post
-
  - eth_getBlockTransactionCountByNumber - Polygon PoS post
-
  - eth_getCode - Polygon PoS post
-

  - eth_getFilterChanges - Polygon PoS post
-
  - eth_getFilterLogs - Polygon PoS post
-
  - eth_getLogs - Polygon PoS post
-
  - eth_getRootHash - Polygon PoS post
-
  - eth_getSignersAtHash - Polygon PoS post
-
  - eth_getStorageAt - Polygon PoS post
-
  - eth_getTransactionByBlockHashAndIndex - Polygon PoS post
-
  - eth_getTransactionByBlockNumberAndIndex - Polygon PoS post
-
  - eth_getTransactionByHash - Polygon PoS post
-
  - eth_getTransactionCount - Polygon PoS post
-
  - eth_getTransactionReceipt - Polygon PoS post
-
  - eth_getTransactionReceiptsByBlock - Polygon PoS post
-
  - eth_sendRawTransaction - Polygon PoS post
-
  - eth_uninstallFilter - Polygon PoS post
-
  - net_listening - Polygon PoS post
-
  - eth_getUncleCountByBlockHash - Polygon PoS post
-
  - eth_getUncleCountByBlockNumber - Polygon PoS post
-
  - eth_newBlockFilter - Polygon PoS post
-
  - eth_newFilter - Polygon PoS post
-
  - eth_newPendingTransactionFilter - Polygon PoS post
-
  - web3_clientVersion - Polygon PoS post
-
  - eth_createAccessList - Polygon PoS post
-
  - eth_blockNumber - Polygon PoS post
-
  - bor_getSignersAtHash - Polygon PoS post
-
  - net_version - Polygon PoS post
-
  - eth_getProof - Polygon PoS post
-
  - eth_getUncleByBlockNumberAndIndex - Polygon PoS post
-
  - eth_subscribe - Polygon PoS
-
  - eth_unsubscribe - Polygon PoS

Polygon zkEVM

- Polygon zkEVM API Quickstart
- Polygon zkEVM API FAQ
-
  - What is the difference between Polygon zkEVM and Ethereum?
-
  - What is the difference between Polygon zkEVM and Polygon PoS?
- Polygon zkEVM Endpoints
-

  - eth_getTransactionCount - Polygon zkEVM post
-
  - eth_call - Polygon zkEVM post
-
  - eth_chainId - Polygon zkEVM post
-
  - eth_newBlockFilter - Polygon zkEVM post
-
  - eth_estimateGas - Polygon zkEVM post
-
  - eth_newFilter - Polygon zkEVM post
-
  - eth_gasPrice - Polygon zkEVM post
-
  - eth_sendRawTransaction - Polygon zkEVM post
-
  - eth_getBalance - Polygon zkEVM post
-
  - eth_uninstallFilter - Polygon zkEVM post
-
  - eth_getBlockByHash - Polygon zkEVM post
-
  - net_version - Polygon zkEVM post
-
  - eth_getBlockByNumber - Polygon zkEVM post
-
  - web3_clientVersion - Polygon zkEVM post
-
  - eth_getBlockTransactionCountByHash - Polygon zkEVM post
-
  - eth_getBlockTransactionCountByNumber - Polygon zkEVM post
-
  - zkevm_batchNumber - Polygon zkEVM post
-
  - eth_getCode - Polygon zkEVM post
-
  - eth_getFilterChanges - Polygon zkEVM post
-
  - eth_getFilterLogs - Polygon zkEVM post
-
  - zkevm_getBatchByNumber - Polygon zkEVM post
-
  - eth_getLogs - Polygon zkEVM post
-
  - zkevm_getBroadcastURI - Polygon zkEVM post
-
  - eth_getStorageAt - Polygon zkEVM post
-
  - zkevm_isBlockConsolidated - Polygon zkEVM post
-
  - eth_getTransactionByBlockHashAndIndex - Polygon zkEVM post
-
  - zkevm_isBlockVirtualized - Polygon zkEVM post
-
  - eth_getTransactionByBlockNumberAndIndex - Polygon zkEVM post
-
  - zkevm_verifiedBatchNumber - Polygon zkEVM post
-
  - eth_getTransactionByHash - Polygon zkEVM post
-
  - zkevm_virtualBatchNumber - Polygon zkEVM post
-
  - eth_getCompilers - Polygon zkEVM post
-
  - eth_getUncleByBlockHashAndIndex - Polygon zkEVM post
-
  - eth_getUncleByBlockNumberAndIndex - Polygon zkEVM post
-

- eth_getUncleCountByBlockHash - Polygon zkEVM post
-
    - eth_getUncleCountByBlockNumber - Polygon zkEVM post
-
    - eth_protocolVersion - Polygon zkEVM post
-
    - eth_blockNumber - Polygon zkEVM post
-
    - eth_getTransactionReceipt - Polygon zkEVM post
-
    - zkevm_batchNumberByBlockNumber - Polygon zkEVM post
-
    - zkevm_consolidatedBlockNumber - Polygon zkEVM post
-
    - zkevm_estimateFee API - Polygon zkEVM post
-
    - zkevm_estimateGasPrice API - Polygon zkEVM post

Arbitrum

- Arbitrum API Quickstart
-
    - Arbitrum SDK Examples
- Arbitrum API FAQ
-
    - Arbitrum vs. Ethereum API Differences
- Arbitrum API Endpoints
-
    - eth_call - Arbitrum post
-
    - eth_estimateGas - Arbitrum post
-
    - eth_accounts - Arbitrum post
-
    - eth_blockNumber - Arbitrum post
-
    - eth_chainId - Arbitrum post
-
    - eth_gasPrice - Arbitrum post
-
    - eth_getBalance - Arbitrum post
-
    - eth_getBlockTransactionCountByHash - Arbitrum post
-
    - eth_getBlockTransactionCountByNumber - Arbitrum post
-
    - eth_getCode - Arbitrum post
-
    - eth_getFilterChanges - Arbitrum post
-
    - eth_getFilterLogs - Arbitrum post
-
    - eth_getLogs - Arbitrum post
-
    - eth_getStorageAt - Arbitrum post
-
    - eth_getTransactionByBlockHashAndIndex - Arbitrum post
-
    - eth_getTransactionCount - Arbitrum post
-
    - eth_getUncleByBlockNumberAndIndex - Arbitrum post
-
    - eth_getUncleCountByBlockHash - Arbitrum post
-
    - eth_getUncleCountByBlockNumber - Arbitrum post
-
    - eth_newBlockFilter - Arbitrum post
-

  - eth_newFilter - Arbitrum post
-
  - eth_newPendingTransactionFilter - Arbitrum post
-
  - eth_uninstallFilter - Arbitrum post
-
  - net_listening - Arbitrum post
-
  - net_version - Arbitrum post
-
  - web3_clientVersion - Arbitrum post
-
  - web3_sha3 - Arbitrum post
-
  - eth_sendRawTransaction - Arbitrum post
-
  - eth_createAccessList - Arbitrum post
-
  - eth_maxPriorityFeePerGas - Arbitrum post
-
  - eth_feeHistory - Arbitrum post
-
  - eth_getBlockByHash - Arbitrum post
-
  - eth_getBlockByNumber - Arbitrum post
-
  - eth_getTransactionByBlockNumberAndIndex - Arbitrum post
-
  - eth_getTransactionByHash - Arbitrum post
-
  - eth_getProof - Arbitrum post
-
  - eth_getTransactionReceipt - Arbitrum post
-
  - eth_getUncleByBlockHashAndIndex - Arbitrum post
-
  - eth_subscribe
-
  - eth_unsubscribe

Optimism

- Optimism API Quickstart
-
  - Optimism SDK Examples
- Optimism API FAQ
-
  - Optimism Error Codes
- Optimism API Endpoints
-
  - eth_call - Optimism post
-
  - eth_estimateGas - Optimism post
-
  - eth_accounts - Optimism post
-
  - eth_blockNumber - Optimism post
-
  - eth_chainId - Optimism post
-
  - eth_gasPrice - Optimism post
-
  - eth_getBalance - Optimism post
-
  - eth_getBlockTransactionCountByHash - Optimism post
-
  - eth_getBlockTransactionCountByNumber - Optimism post
-

  - eth_getCode - Optimism post
-
  - eth_getFilterChanges - Optimism post
-
  - eth_getFilterLogs - Optimism post
-
  - eth_getLogs - Optimism post
-
  - eth_getStorageAt - Optimism post
-
  - eth_getTransactionByBlockHashAndIndex - Optimism post
-
  - eth_getTransactionByBlockNumberAndIndex - Optimism post
-
  - eth_getTransactionByHash - Optimism post
-
  - eth_getTransactionCount - Optimism post
-
  - eth_getTransactionReceipt - Optimism post
-
  - eth_getUncleByBlockHashAndIndex - Optimism post
-
  - eth_getUncleByBlockNumberAndIndex - Optimism post
-
  - eth_getUncleCountByBlockHash - Optimism post
-
  - eth_getUncleCountByBlockNumber - Optimism post
-
  - eth_newBlockFilter - Optimism post
-
  - eth_newFilter - Optimism post
-
  - eth_newPendingTransactionFilter - Optimism post
-
  - eth_protocolVersion - Optimism post
-
  - eth_sendRawTransaction - Optimism post
-
  - eth_syncing - Optimism post
-
  - eth_uninstallFilter - Optimism post
-
  - net_listening - Optimism post
-
  - net_version - Optimism post
-
  - web3_clientVersion - Optimism post
-
  - web3_sha3 - Optimism post
-
  - eth_getBlockByHash - Optimism post
-
  - eth_getBlockByNumber - Optimism post
-
  - eth_getProof - Optimism post
-
  - eth_subscribe
-
  - eth_unsubscribe

Base

- Base API Quickstart
- Base API FAQ
- Base API Endpoints
-
  - eth_accounts - Base post
-

  - eth_blockNumber - Base post
-
  - eth_call - Base post
-
  - eth_chainId - Base post
-
  - eth_estimateGas - Base post
-
  - eth_feeHistory - Base post
-
  - eth_gasPrice - Base post
-
  - eth_getBalance - Base post
-
  - eth_getBlockByHash - Base post
-
  - eth_getBlockByNumber - Base post
-
  - eth_getBlockTransactionCountByHash - Base post
-
  - eth_getBlockTransactionCountByNumber - Base post
-
  - eth_getCode - Base post
-
  - eth_getFilterChanges - Base post
-
  - eth_getFilterLogs - Base post
-
  - eth_getLogs - Base post
-
  - eth_getProof - Base post
-
  - eth_getStorageAt - Base post
-
  - eth_getTransactionByBlockHashAndIndex - Base post
-
  - eth_getTransactionByBlockNumberAndIndex - Base post
-
  - eth_getTransactionByHash - Base post
-
  - eth_getTransactionCount - Base post
-
  - eth_getTransactionReceipt - Base post
-
  - eth_getUncleByBlockHashAndIndex - Base post
-
  - eth_getUncleByBlockNumberAndIndex - Base post
-
  - eth_getUncleCountByBlockHash - Base post
-
  - eth_getUncleCountByBlockNumber - Base post
-
  - eth_maxPriorityFeePerGas - Base post
-
  - eth_newBlockFilter - Base post
-
  - eth_newFilter - Base post
-
  - eth_newPendingTransactionFilter - Base post
-
  - eth_protocolVersion - Base post
-
  - eth_sendRawTransaction - Base post
-
  - eth_syncing - Base post
-
  - eth_uninstallFilter - Base post
-

  - net_listening - Base post
-
  - web3_sha3 - Base post

❇ Solana

- Solana API Quickstart
- Solana API FAQ
- Solana API Endpoints
-
  - getAccountInfo post
-
  - simulateTransaction post
-
  - getBalance post
-
  - getBlock post
-
  - getBlockCommitment post
-
  - getBlockProduction post
-
  - getBlocks post
-
  - getBlocksWithLimit post
-
  - getBlockTime post
-
  - getClusterNodes post
-
  - getEpochInfo post
-
  - getEpochSchedule post
-
  - getFeeForMessage post
-
  - getFirstAvailableBlock post
-
  - getGenesisHash post
-
  - getHealth post
-
  - getHighestSnapshotSlot post
-
  - getIdentity post
-
  - getInflationGovernor post
-
  - getInflationRate post
-
  - getInflationReward post
-
  - getLargestAccounts post
-
  - getMaxRetransmitSlot post
-
  - getMaxShredInsertSlot post
-
  - getMinimumBalanceForRentExemption post
-
  - getMultipleAccounts post
-
  - getProgramAccounts post
-
  - getRecentPerformanceSamples post
-
  - getSignaturesForAddress post
-

  - getSignatureStatuses post
-
  - getSlot post
-
  - getSlotLeader post
-
  - getSlotLeaders post
-
  - getSupply post
-
  - getTokenAccountBalance post
-
  - getTokenAccountsByOwner post
-
  - getTokenSupply post
-
  - getTransaction post
-
  - getVersion post
-
  - getVoteAccounts post
-
  - isBlockhashValid post
-
  - minimumLedgerSlot post
-
  - sendTransaction post
-
  - requestAirdrop post
-
  - getBlockHeight post
-
  - getRecentBlockhash post

Astar

- Astar API Quickstart
- Astar API FAQ
- Astar API Endpoints
-
  - eth_accounts - Astar post
-
  - eth_getTransactionReceipt - Astar post
-
  - eth_maxPriorityFeePerGas - Astar post
-
  - eth_blockNumber - Astar post
-
  - eth_call - Astar post
-
  - eth_chainId - Astar post
-
  - eth_gasPrice - Astar post
-
  - eth_getBalance - Astar post
-
  - eth_getBlockByHash - Astar post
-
  - eth_getBlockByNumber - Astar post
-
  - eth_getBlockTransactionCountByHash - Astar post
-
  - eth_getBlockTransactionCountByNumber - Astar post
-
  - eth_getCode - Astar post
-
  - eth_getStorageAt - Astar post
-

  - eth_getTransactionByBlockHashAndIndex - Astar post
-
  - eth_getTransactionByBlockNumberAndIndex - Astar post
-
  - eth_getTransactionByHash - Astar post
-
  - eth_getTransactionCount - Astar post
-
  - eth_getUncleByBlockNumberAndIndex - Astar post
-
  - eth_sendRawTransaction - Astar post
-
  - net_version - Astar post
-
  - web3_clientVersion - Astar post
-
  - web3_sha3 - Astar post
-
  - eth_getLogs - Astar post
-
  - eth_getFilterChanges - Astar post
-
  - eth_getFilterLogs - Astar post
-
  - eth_newFilter - Astar post
-
  - eth_newPendingTransactionFilter - Astar post
-
  - eth_uninstallFilter - Astar post
-
  - eth_newBlockFilter - Astar post
-
  - eth_estimateGas - Astar post
-
  - eth_subscribe
-
  - eth_unsubscribe

STARKNET

- Starknet API Quickstart
- Starknet API FAQ
- Starknet API Endpoints
-
  - starknet_addDeclareTransaction post
-
  - starknet_getClassAt post
-
  - starknet_addDeployAccountTransaction post
-
  - starknet_getClassHashAt post
-
  - starknet_addInvokeTransaction post
-
  - starknet_getEvents post
-
  - starknet_blockHashAndNumber post
-
  - starknet_getNonce post
-
  - starknet_blockNumber post
-
  - starknet_getStateUpdate post
-
  - starknet_call post
-
  - starknet_getStorageAt post
-

  - starknet_chainId post
-
  - starknet_getTransactionByBlockIdAndIndex post
-
  - starknet_estimateFee post
-
  - starknet_getTransactionByHash post
-
  - starknet_getBlockTransactionCount post
-
  - starknet_getTransactionReceipt post
-
  - starknet_getBlockWithTxHashes post
-
  - starknet_pendingTransactions post
-
  - starknet_getBlockWithTxs post
-
  - starknet_syncing post
-
  - starknet_getClass post
-
  - starknet_estimateMessageFee post

NFT API FAQ

Frequently Asked Questions regarding our NFT API

What can I make with the NFT API?

Using the Alchemy NFT API lets you fetch and display NFTs for your users, making it easy to build all kinds of NFT projects. For a list of copy-able scripts that use the NFT API to fetch NFT data, check out theRecipes hub!

Tell us onTwitter what you're trying to build!

What do NFT API requests look like?

We've compiled some super easy to use, copy-and-paste code snippets

You can check out the rest of our docs for more in-depth documentation about our endpoints, but for a quick walkthrough with some sample code ready to run, start with this GitHub repository!

github.com GitHub - alchemyplatform/nft-api-javascript-scripts: A collection of Javascript scripts running with Alchemy Web3.js, Fetch, or Axios

Which Chains and Networks are Supported?

For a full list of supported chains per NFT API method check out theNFT API Endpoints page .

Currently Supported Chains

Ethereum

→ Mainnet, Goerli, Sepolia

Polygon (Matic)

→ Mainnet, Mumbai

Arbitrum

→ Mainnet, Goerli, Sepolia

Optimism

→ Mainnet, Goerli, Sepolia

Base

→ Mainnet, Goerli

Starknet

→ Mainnet, Goerli

What types of NFTs are supported?

All NFTs made with the ERC721 and ERC1155 standards are supported by the NFT API. At the moment, we support a select number of tokens like CryptoPunks and CryptoKitties that pre-date the existence of standardized NFT contracts.

Alchemy is actively working on adding support for as many blockchains / NFT standards as we can. To influence our roadmap and let your voice be heard, please upvote or add requests on our publicfeedback tracker .

Missing NFTs?

If you find that NFTs are missing in wallets you are calling, please provide details about the missing NFThere so we can get this resolved for you!

How does Alchemy determine NFT Standard i.e. if a collection is

ERC721 or ERC1155 ?

First, we collect the following data (each of which is an eth_call):

- Does the contract have any code associated with it, i.e. is it even a contract at all?
- Does the contract support the
- erc721
- interface?
- Does the contract support the
- erc1155
- interface?
- Does the contract have any
- erc721
- transfers?
- Does the contract have any
- erc1155
- transfers?

After we've collected the data, we have some simple code to determine token standard

1. If a contract looks like it supports both the
2. erc721
3. and
4. erc1155
5. standards, we return a token type of
6. erc721
7. We first query
8. supportsInterface
9. and then query
10. hasTransfers
11. . The standards are pretty clear that the contract should return
12. true
13. for a
14. supportsInterface
15. call, so we look at that first.
16. Finally, we have two token types:
17. NotAContract
18. and
19. NoSupportedNFTStandard
20. . These token types communicate that we are, in fact, aware of the contract and we are telling you that it is not an NFT

contract.

How does NFT Metadata Work?

The primary object within the Alchemy NFT API is theNFT asset , which is an on-chain item that lives on the blockchain. Within each NFT, there can be many different fields that describe its on-presence.

General Metadata Structure

image A URL to the NFT asset image. Can be standard URLs pointing to images on conventional servers,IPFS , orArweave . Most types of images (SVGs, PNGs, JPEGs, etc.) are supported by NFT marketplaces. external_url The image URL that appears alongside the asset image on NFT platforms. It tends to be the full-size, highest resolution media file. background_color Background color of the NFT item. Usually must be defined as a six-character hexadecimal. name Name of the NFT asset. description A human-readable description of the NFT asset. (Markdown is supported/rendered on OpenSea and other NFT platforms) attributes The traits/attributes/characteristics for each NFT asset.

Missing Metadata Fields?

Not all metadata fields may be filled out by the NFT creator. NFTs can be published on-chain without these fields and still conform to the ERC721/ERC1155 standards.

What are Gateway vs. Raw URIs?

Gateways are an important part of NFT infrastructure. Behind the scenes, they allow users to access IPFS content without running an IPFS node. With a gateway provider, a third-party service downloads data off of IPFS nodes and then serves it whenever requested.

NFT creators/developers can also "pin" their content, effectively caching and storing it on gateway nodes/servers. This ensures that the content is always available online.

While NFT gateway and raw URIs tend to be pointed at the same links, gateway URIs generally offer better performance.

What is the difference between

getNFTsForOwner & getNFTMetadata ?

getNFTsForOwner is most commonly used when querying all NFTs owned by an address. By default, it will return both NFTs and any associated metadata per asset in the response. Common use cases include dashboards/wallets for viewing NFT assets held by a particular address.

getNFTMetadata is more specific and is used for querying the metadata of a single NFT. Common use cases include NFT rarity tools and NFT searching applications.

How are NFTs Classified as "Spam"?

Given a contract address, we look at a few things:

1. If this contract is ERC721, does this contract egregiously break the ERC721 standard? i.e., Does it have a lot of duplicate tokens.
2. If this contract is ERC721, does it have any transfer during which it broke the ERC721 standard? i.e. It transferred a token to more than one recipient.
3. Does this contract mint tokens mostly to honeypots? Honeypots are popular addresses like
4. vitalik.eth
5. Does this contract egregiously lie about its own total supply? i.e. running
6. totalSupply()
7. on the contract is vastly different from the empirical number of tokens in circulation.

If any of these are satisfied, we will mark an NFT as spam.

How can I understand why a particular NFT collection is marked as "Spam"?

The response object of Alchemy's getNFTs method contains a field called classifications inside spamInfo . This field is a list of tags, each of which map to a reason why the NFT was considered spam.

The tags are one of the below :-

1. Erc721TooManyOwners
2. : A single token in the ERC721 collection has been transferred to multiple owners indicating fraudulent transfers
3. Erc721TooManyTokens
4. : This collection has a lot of duplicate tokens indicating spammy behaviour
5. Erc721DishonestTotalSupply
6. : The contract lies about its own token supply. Running
7. totalSupply()
8. on the contract is vastly different from the empirical number of tokens in circulation.
9. MostlyHoneyPotOwners
10. : Most or all of the owners of the NFT collection are Honeypots i.e. popular addresses like
11. vitalik.eth
12. . So these are probably unwarranted airdrops.
13. OwnedByMostHoneyPots
14. : A significant chunk of the usual Honeypot addresses own this collection. Again signifying unwarranted airdrops.

How often is the floor price updated for

getFloorPrice ?

Floor price is queried per collection at request time and cached for 5 minutes. After that the floor price is refreshed by a best-effort fetch from the marketplace. The retrievedAt field is the timestamp of when the collection floor price was last updated for a marketplace.

Why does the

getFloorPrice endpoint return slightly older floor price?

The cache for the getFloorPrice function is only refreshed every 15 mins. So if your transaction has happened in the last 15 mins, you might want to wait to get the data.

NFT Images and Media FAQ

Why does Alchemy Cache NFT Media?

While NFT media is traditionally served from IPFS/third-party servers, developers often face slow loading times and timeout errors when using these endpoints. Alchemy solves this problem by caching NFT images and serving up NFT URLs from our own cache.

How do I use Alchemy-hosted NFT media?

Where available, Alchemy will replace the default gateway field of the NFT media object with the Alchemy NFT-CDN URL.

Alchemy-hosted NFT Asset

Third-party hosted NFT Asset

Here's an example of a BAYC NFT asset cached by Alchemy:https://nft-cdn.alchemy.com/eth-mainnet/d9d674adddb2da8fcbcdb0230ca1da3d

Why did my request not return a cached media url?

We're actively working on expanding coverage of Alchemy-hosted NFT endpoints. Feel free to reach out tosupport@alchemy.com for specific questions!

Which media URL should I use to get the NFT image?

When displaying the media associated with an NFT, it is recommended to use the media.gateway URL as the preferred media URL. If this URL is not available, fall back to media.raw . If neither is available, use metadata.image .

How does image resizing work?

Currently, NFT assets are now stored on Alchemy's cloud/CDN (nft-cdn.alchemy.com). We only use Cloudinary for transformations (like resizing, image format conversion, etc.).

Since we automatically generate thumbnails for a cached asset in the payload response, we recommend users use that.

However, we still enable resizing, but not by default. For example, if you take a BAYC stored on our CDN https://nft-cdn.alchemy.com/eth-mainnet/d9d674adddb2da8fcbcdb0230ca1da3d .

Base Cloudinary URL:

The thumbnail will be located at a corresponding Cloudinary URL with the same hash:https://res.cloudinary.com/alchemyapi/image/upload/thumbnail/eth-mainnet/d9d674adddb2da8fcbcdb0230ca1da3d . Developers can then request resized images for thumbnails, smartphones, tablets, and/or laptop viewing, you can manipulate the URL to include height/width like before:

Base Cloudinary URL with Width & Height Values:

https://res.cloudinary.com/alchemyapi/image/upload/w_400,h_400/thumbnail/eth-mainnet/d9d674adddb2da8fcbcdb0230ca1da3d

Note that this resizes the thumbnail version.

Create your own size

Developers can use any combination of width and height pixel values to create the desired aspect ratio for visual displays. Here's an example of a resized BAYC NFT asset cached by Alchemy: https://res.cloudinary.com/alchemyapi/image/upload/w_400,h_400/thumbnail/eth-mainnet/d9d674adddb2da8fcbcdb0230ca1da3d

How can I get PNG Images instead of SVG when using Alchemy's APIs?

By default Alchemy API returns the same image format as is in the metadata. Sometimes that means SVG images. In case you want to obtain the PNG format for those images instead, you can replace the string /thumbnail with /convert-png in the thumbnail URL value.

Example: If the thumbnail URL is https://res.cloudinary.com/alchemyapi/image/upload/thumbnail/eth-mainnet/ , you should replace it with https://res.cloudinary.com/alchemyapi/image/upload/convert-png/eth-mainnet/ to get the PNG image

How can I resize a thumbnail and maintain the original resolution and aspect ratio?

In case you want to resize the media, you can make alterations to the original thumbnail URL or image URL to get the new URL that will return you a resized thumbnail with maintained aspect ratio.

Example: If the thumbnail URL is https://res.cloudinary.com/alchemyapi/image/upload/thumbnail/eth-mainnet/ , you should replace it with https://res.cloudinary.com/alchemyapi/image/upload/w_256/scaled/eth-mainnet/ to get the resized image with width 256 and same aspect ratio

Handling NFT API Errors

While we work to ensure that every NFT's metadata is returned when requested from the API, there are various reasons why we may not able to fulfill your request.

If present, errors in metadata fulfillment will appear in the error field of the response payload. The following is a list of errors which are "retryable" — requests that receive these errors have a strong chance of succeeding on a repeated attempt:

- Token uri responded with a non 200 response code
- Throttled token uri
- IPFS gateway timed out
- Centralized gateway timed out, try again with a higher tokenUri timeout
- ArWeave gateway timed out
- Internal service

Below are more detailed breakdowns of what many of our errors mean and how to approach them.

Token does not exist

Composer Example

Contract Address = 0x60e4d786628fea6478f785a6d7e704777c86a7c6 Token ID = 2079999

Why?

In order to fetch the metadata for a given NFT, we call one of two potential methods on the contract with the token ID as the input. For ERC721 contracts, we call the tokenURI method. For ERC1155 contracts, we call uri . These two methods take in a token ID and return a uri which points to the metadata for that token ID.

If we pass a token ID into the method that the contract does not recognize, we will get one of several errors that each mean that the token does not exist. Essentially, the contract does not recognize the token ID that you provided.

See the specific error for the example above via Etherscan's contract interface.

The contract throws an exception on etherscan

Next Steps

More often than not, a contract will return a "token does not exist" error when the token isnot yet minted . It's important to note that the token ID may, however, exist at a later date once it is minted. If you are confident that the token ID you providedshould exist, you can retry your query at a later date. Once the token has been minted and recognized by our service, your request may succeed. Note that some contracts do return un-minted tokens, so this error message is not to be used as an indicator for whether a token has been minted yet.

The second most common reason for a "token does not exist" error is that the token truly does not exist! For older contracts that have been completely minted, this error should be trusted. Repeated attempts at finding metadata will fail simply because it does not exist!

Malformed token URI

Composer Example

Contract Address: 0xbfde6246df72d3ca86419628cac46a9d2b60393c Token ID: 14506

Why?

Once we know where an NFT's metadata is stored (by calling tokenURI or uri on the contract as described above), we visit the resulting website in order to access the metadata. However, if the website that is returned by tokenURI or uri is malformed, then we cannot visit it and return this error instead. By "malformed" we mean any website that cannot be visited. In the example above (click on it to see) you can see that the tokenUri.raw field is an empty string. That is because the tokenURI method of that contract returned an empty string instead of a valid website.

You can see the empty response in etherscan's contract interface here:https://etherscan.io/address/0xbfde6246df72d3ca86419628cac46a9d2b60393c#readContract

The contract returns a malformed URI on etherscan

Next Steps

Unfortunately, there isn't much you can do here. If you are the project owner or you happen to know a special URI that Alchemy should return for a contract that isnot included in the tokenURI or uri methods, then let us know!

Failed to get token uri

Composer Example

Contract Address: 0xffdf17652cca46eb98a214cb3e413c8661241e49

Token ID: 7818

Why?

In the "token does not exist" section we talked about how a contract can throw an exception when we ask for the tokenURI . If the exception indicates that the token does not exist, then we return the "token does not exist" error. If the exception isany other type then we return the generic "Failed to get token uri" error.

You can see the specific error for the example above in etherscan's contract interface here:https://etherscan.io/address/0xffdf17652cca46eb98a214cb3e413c8661241e49#readContract

The contract throws an exception on etherscan

Next Steps

Unfortunately, there isn't much you can do if the contract doesn't properly return a token URI. It ispossible that there was a transient error running an eth_call on our nodes, but it's pretty unlikely. Feel free to retry the request!

Token URI responded with a non-200 response code

Composer Example

Contract Address: 0x909899c5dbb5002610dd8543b6f638be56e3b17e

Token ID: 955 Why?

In the section above we talked about how we get the URI where the NFT metadata lives. Once we have the URI, we then attempt to visit it in order to access the metadata. If the URI responds with anything other than a 2xx response code, like for instance a 502 Bad Gateway (thePlasmaBear contract above is an example) then we return this error.

Next Steps

In this case, it ispossible that retrying the request can succeed. If the contract's metadata website is down for some transient reason then a retry could work. A more common case is that the website may be rate-limiting Alchemy servers and returning 4xx . We are working on infrastructure to reduce the occurrence of this error. In the meantime, we suggest retries with a reasonable backoff strategy.

Throttled token URI

Why?

If a token URI containing metadata responds to our retrieval attempts with a "429 Too Many Requests," the site is informing us that we have requested metadata too often. When this happens, we will not visit the website again in order to release the "rate limit" that they have put on us. During this "waiting period," requesting NFT metadata for that same asset requires hitting that website and, therefore, we will temporarily block the request.

Next Steps

In this case, you should retry your request after a variable number of seconds. (We suggest waiting at least 10 seconds) If you continue to be throttled, increase the waiting period a bit longer after each retry.

Contract does not have any code

Why?

Not all addresses are token contracts! If you send us an address for which there is no contract code then we return this error. In order to determine if the address is a contract, we call eth_getCode on the address.

Next steps

You should not retry this request. Perhaps you are accessing the contract on the wrong network. For instance, you might want to find the contract on Polygon rather than Ethereum.

Contract returned a broken token URI, do not retry

Why?

This occurs when the tokenUri associated with the NFT does not respond to the web request at the time the request was made. This can happen for many reasons including the url not existing, being deprecated, or lacking DNS set up.

This can also happen if the content length of the response is larger than 30 000 bytes.

Next steps

There is a chance that the URL gets fixed, in which case retrying the request will return the updated tokenUri if the URL has been updated in time. Otherwise, the URL may be permanently dead.

Bad Token URI

Why?

This error occurs when Alchemy has tried to fetch data from a tokenUri multiple times but the fetch query always either times out or returns unrecoverable errors repeatedly. This can happen due to multiple reasons including the server being unreachable or lacking DNS set up.

Next steps

This is often an issue with the NFT project itself and it may be worth checking if the corresponding NFT project is still alive or dead. In case it is still alive, we recommend you report the issue to the NFT project and ask them to update the tokenUri returned by their smart contract.

Updated 5 months ago

What’s Next Have any other questions about the NFT API? Checkout and post a question to our discussion forum:https://docs.alchemy.com/discuss Did this page help you?Yes No