midnight

Scavenger Mine API reference

October 2025 / Version 1.0

Legal disclaimer

The information provided herein is for informational purposes only and should not be construed as financial, legal, or investment advice. We and our affiliates do not recommend that NIGHT or DUST or any digital assets be bought, sold, swapped, staked, or held by you. It is the responsibility of any person who accesses the information herein to observe all applicable laws and regulations of their relevant jurisdiction. By proceeding to obtain the information, you are representing and warranting that all the applicable laws and regulations of your jurisdiction allow you to access such information. We and our affiliates make no representations or warranties of any kind, express or implied, regarding the accuracy, completeness, or reliability of the information contained herein. We and our affiliates assume no liability for any losses or damages that may result from reliance on the information contained in this document. This document may contain forward-looking statements that are subject to risks and uncertainties that could cause actual results to differ materially from those expressed or implied by such statements. Forward-looking statements in this document represent our judgment as of the date of the document. We do not undertake any obligation to update or revise any forward-looking statement to reflect new information or future events. You should not place undue reliance on forward-looking statements contained in this document.

The Scavenger Mine program is a 21-day proof-of-work system designed so that anyone with an ordinary computer may participate, without requiring specialized hardware. Participants only need to register once (per Destination address) by accepting the Token End User Terms, after which the Scavenger Mine process runs automatically on a browser app (or alternatively, via CLI/scripts and using the API whose references are provided below).

Purpose

This document describes the API calls available to anyone who wishes to participate in Scavenger Mine without going through the browser interface.

Audience

This document is technical in nature and is intended for use by people who are skilled in software development and have a good understanding of blockchain and cryptographic principles. For non-technical users, the use of the browser interface is encouraged.

System overview

- Produces a new challenge every 60 mins for 21 days (24 different challenges each day - 504 in all).
- Solutions may be submitted up to 24hrs after first availability (with the exception of the final day, when all solutions must be submitted before the phase closes).
- Participants must include their Destination address when constructing the preimage of their solution to each challenge.
- Destination addresses must be registered prior to solutions being accepted.
- Destination addresses may be registered starting 24 hours before Scavenger Mine begins and throughout the whole period.
- Each address registration represents a signed acceptance of the published Token End User Agreement.
- The server responds to good solutions with a signed receipt.
- The server creates a Merkle tree of daily receipts and submits the root to Cardano mainnet.
- The server updates NIGHT allocations daily, based on the daily Merkle tree.
- The daily Merkle root is used as proof of no-pre-mine during the following day. The fact that this is unpredictable (even to the operators) ensures fairness.
- Address pre-registrations are used as source of randomness for proof of no-pre-mine on the first day.
- The server provides an endpoint to allow participants to consolidate rewards earned through one address, to be redeemed through another address.
    - Consolidation (referred to as donation) may be performed throughout the Scavenger Mine phase and for 24 hours after the phase closes.

API calls

Diagram 1 shows the expected order of API calls that a participant will need to call in order to fetch challenges and submit solutions. Four endpoints in the diagram are necessary (/TandC, /register, /challenge and /solution), the others (/work_to_star_rate and /donate_to) are optional.

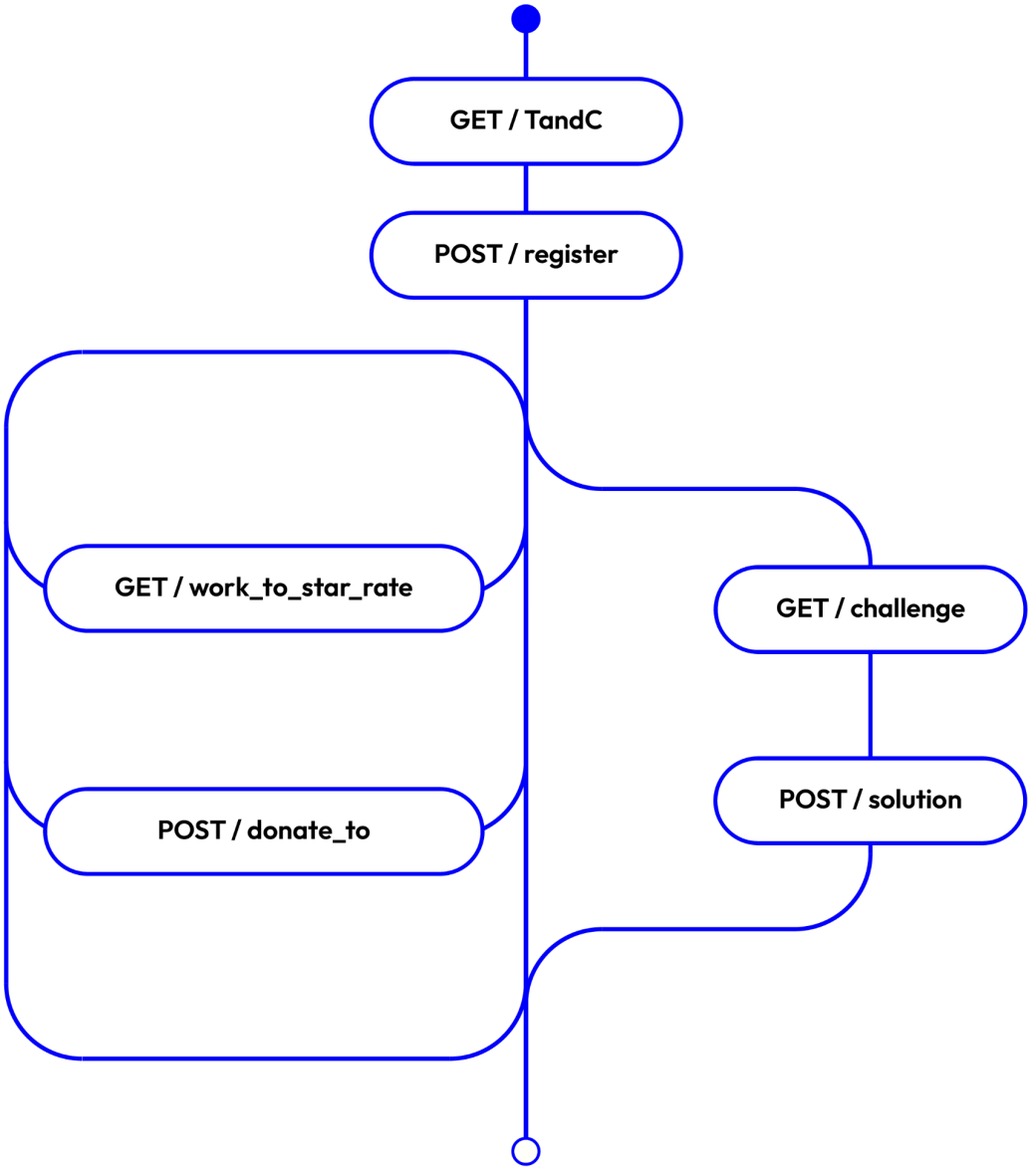

Diagram 1: general flow of API calls

GET /TandC - Obtain the Token End User Agreement (terms and conditions)

Endpoint GET	/TandC/{version}	
Public access	Yes	
Authentication	No	
Availability days	0, 1-21, 22	
Parameters	Version (optional: default 1-0)	
Response JSON object	version	String: "1-0"
	content	String: "**TOKEN END-USER TERMS**\n\nLast updated: 15 Aug 2025.\n\nThese Token End-User Terms ...
	message	String: "I agree to abide by the terms and conditions as described in version 1-0 of the Midnight scavenger mining process: 281ba5f69f4b943e3fb8a20390878a232787a04e4be22177f2472b63df01c200"
Curl	curl https://scavenger.prod.gd.midnighttge.io/TandC	
Example response	{ "version": "1-0", "content": "**TOKEN END-USER TERMS**\n\nLast updated: 15 Aug 2025.\n\nThese Token End-User Terms (the “**Token End-User Terms**”) ...", "message": "I agree to abide by the terms and conditions as described in version 1-0 of the Midnight scavenger mining process: 281ba5f69f4b943e3fb8a20390878a232787a04e4be22177f2472b63df01c200" }	
Response when version string is added to the request (e.g. TandC/2-0)	{ "message": "Terms and Conditions version 2-0 not found", "error": "Not Found", "statusCode": 404 }	

POST /register - Register a Destination address to participate in Scavenger Mine

Endpoint POST	/register/{address}/{signature}/{pubkey}	
Public access	Yes	
Authentication	No	
Availability days	0, 1-21, 22	
Parameters	address	String: Standard Cardano payment address
	signature	String: Standard CIP 8/30 signature (see preimage below)
	pubkey	String: 64 character hex encoded - associated with the address.
Response	registration_receipt JSON object	

Notes and example	**Message (exactly as returned from GET /TandC):**
	`I agree to abide by the terms and conditions as described in version 1-0 of the Midnight scavenger mining process: 281ba5f69f4b943e3fb8a20390878a232787a04e4be22177f2472b63df01c200`
	
	Address:
	`addr1qqwzupt3gvehw9s92w2qwsjkk7wl0lpkq5vq9n80cxed80e2wumjk88lz63vlc0f5c6tl2hrca8geqvguczr74ezjhcq2x66y3`
	
	PubKey:
	`009236f53f5fbb1056defb64d623f7508bc1b61822016d43faf62070f1489ff5`
	
	Signature:
	`845882a30127045839001c2e0571433377160553940742 56b79df7fc36051802ccefc1b2d3bf2a77372b1cff16a2cfe1e9a634bfaae3c74e8c8188e6043f572295f0676164647265737358390 01c2e0571433377160553940742 56b79df7fc36051802ccefc1b2d3bf2a77372b1cff16a2cfe1e9a634bfaae3c74e8c8188e6043f572295f0a166686173686564f458b349206167726565207 46f20616269646520627920746865207465726d7320616e6420636f6e646974696f6e732061732064657363726962656420696e2076657273696f6e203 12d30206f6620746865204d69646e6967687420736176656e676572206d696e696e672070726f636573733a20666566653336626638653566623436313663633536386138643762613230616237306361626632653837623866 3661656362393662303264383365643438665840504a01e23e3a6cefcb93901af88f9873421fa76f75f98d7d0c7b43b3bdf09676921d7ee326f3bf1061fea4c62e07b84b9fdd1e17cd5d52790a7c1a0fce99e80e`	

External validation	The signature may be checked at the following web page:	

	https://verifycardanomessage.cardanofoundation.org/ However, this site expects the long-form version of the pubkey, which is available from the Lace wallet and most other wallets.
Curl	`curl -X POST https://scavenger.prod.gd.midnighttge.io/register/addr1qqwzupt3gvehw9s92w2qwsjkk7wl0lpkq5vq9n80cxed80e2wumjk88lz63vlc0f5c6tl2hrca8geqvguczr74ezjhcq2x66y3/845882a30127045839001c2e05714333771605539407425 6b79df7fc36051802ccefc1b2d3bf2a77372b1cff16a2cfe1e9a634bfaae3c74e8c8188e6043f572295f06761646472657373583900 1c2e05714333771605539407425 6b79df7fc36051802ccefc1b2d3bf2a77372b1cff16a2cfe1e9a634bfaae3c74e8c8188e6043f572295f0a166686173686564f458b34920616772656520746f20616269646520627920746865207465726d7320616e6420636f6e646974696f6e732061732064657363726962656420696e2076657273696f6e20312d30206f6620746865204d69646e6967687420736361766 56e676572206d696e696e672070726f636573733a206665666533366266386535666234363136636335363861386437626132306162373063616266326538376238663836616563623936623032643833656434386665840504a01e23e3a6cefcb93901af88f9873421fa76f75f98d7d0c7b43b3bdf09676921d7ee326f3bf1061fea4c62e07b84b9fdd1e17cd5d52790a7c1a0fce99e80e/009236f53f5fbb1056defb64d623f7508bc1b61822016d43faf62070f1489ff5 -d "{}"`

Examples of error responses	**When long form pubkey is used**
	`{ "message": "Invalid pubkey format - must be 64-character hex string", "error": "Bad Request", "statusCode": 400 }`
	
	When incorrect network is used
	`{ "message": "CIP-30 signature verification failed: Wrong network: expected preprod, got mainnet", "error": "Bad Request", "statusCode": 400 }`
	
	When the signature is over an incorrect message
	`{ "message": "CIP-30 signature verification failed: Message in signature does not match provided message", "error": "Bad Request", "statusCode": 400 }`

Example of a successful response	**When registration is successful**
	{
	"registrationReceipt": {
	"preimage":
	"addr1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde8u
	cclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he845882a30127045839
	002adfc6771bfa6e0d1818793298817d42d50d4d77286f673dd94e18334db
	9bedc9f98c7d8a0d757dbfde6c2ac167e9a53544a236278b5e5c667616464
	726573735839002adfc6771bfa6e0d1818793298817d42d50d4d77286f673
	dd94e18334db9bedc9f98c7d8a0d757dbfde6c2ac167e9a53544a236278b5
	e5c6a166686173686564f458b34920616772656520746f206162696465206
	279207468652074657726d7320616e6420636f6e646974696f6e7320617320
	646573637269626564206696e2076657273696f6e20312d30206f662074686
	5204d69646e6967687420736176656e676572206d696e696e672070726f
	636573733a20666566533366266386535666234363136636335363861386
	437626132306162373063616266326538376238663836616563623936623
	032643833656434386658400 97ce058015c51289afce4d565a7696b1a5a1e6
	f69be8e3c644d429a605ce3d5b827c71ebf21dc6613e7937dd34c5697af7c
	bbfd00681bb61c15d6f81c622704cb0ede8ac38867df1a94171e4b1be7c2d
	bc9f33b0879f5833e39fc2279772a04",
	"signature":
	"8d622d59ef73f49c16d627994dfde8f7632fde5d064c32acfde4a5427f5d
	10dd37ba4ce5da48cfce351202ab18c1731c31319ade6d817819866073db4
	0d9fd0c",
	"timestamp": "2025-10-29T20:45:35.425Z"
	}
	}

registrationReceipt	JSON object
preimage	The concatenation of the address that was registered, the wallet's signature over the T&C message (as returned from GET /TandC) and the timestamp. This receipt may be used as proof that an address has been successfully registered.
signature	The Scavenger server's signature over the preimage.
timestamp	The time at which the signature was created.

GET /challenge - Fetch the next available challenge

Endpoint GET	/challenge	
Public access	Yes	
Authentication	No	
Availability days	1-21	
Parameters	none	
Response JSON object	Response object	JSON
	code	String One of three values: `"before"` before active mining begins `"active"` during active mining `"after"` after active mining
	challenge	challenge JSON object, containing:
	challenge_id	String: Used to identify the challenge and also in the preimage. Example for Day 1 and Challenge 24: `"**D01C24"` The "**" is used as a separator in the cryptoReceipt.
	difficulty	String - Hex encoded 4 byte bit mask. Each zero bit of the mask corresponds to a zero prefix bit of a successful AshMaize hash. Also used in the preimage. Example: `"0000007f"`
	no_pre_mine	String: Used to initialize the AshMaize ROM and in the preimage. Example: `"cddba7b592e3133393c16194fac7431abf2f5485ed711db282183c819e08eba9"`
	no_pre_mine_hour	String: Used in the preimage. Example: `"416194743"`

	latest_submission	ISO 8601 date: Any solution submitted after this date and time will NOT be accepted. Example: `"2025-10-30T05:59:59Z"`
	challenge_number	Integer: 1-504
	day	Integer: 1-21
	issued_at	ISO 8601 date
	mining_period_ends	ISO 8601 date: The date and time when all Scavenger mining will stop. All solutions must be submitted before thai time. Example: `"2025-11-20T23:59:59Z"`
	max_day	Integer: 21
	total_challenges	Integer: 1 - 504
	current_day	Integer: 1 - 21
	next_challenge_starts_at	ISO 8601 date Example: `"2025-11-20T20:00:00Z"`
Curl	`curl https://scavenger.prod.gd.midnighttge.io/challenge`	
Example **correct** response Note: Expected size is about 554 bytes	`{ "code": "active", "challenge": { "challenge_id": "**D21C10", "day": 21, "challenge_number": 10, "issued_at": "2025-11-20T11:00:00.000Z", "latest_submission": "2025-11-20T23:59:59Z", "difficulty": "0000FFFF", "no_pre_mine": "cddba7b592e3133393c16194fac7431abf2f5485ed711db282183c819e08ebaa", "no_pre_mine_hour": "548571128" }, "mining_period_ends": "2025-11-20T23:59:59Z", "max_day": 21, "total_challenges": 504, "current_day": 21, "next_challenge_starts_at": "2025-11-20T20:00:00Z" }`	
Unavailable days	Return on day 0	

	<pre>{
	  "code": "before",
	  "starts_at": "2025-10-30T00:00:00.000Z"
	}</pre>Return on day 22<pre>{
	  "code": "after"
	}</pre>

POST /solution - Submit a solution to a challenge

Endpoint POST	/solution/{address}/{challenge_id}/{nonce}	
Public access	Yes	
Authentication	No	
Availability days	1-21	
Parameters	address	String: Standard Cardano address that has previously been registered using POST /register
	challenge_id	String: Exactly as supplied by GET /challenge
	nonce	String: Hex encoded 64 bit number (16 characters) [See below: How to create a nonce?]
Curl	<pre>curl -X POST https://scavenger.prod.gd.midnighttge.io/solution/addr_test1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde8ucclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he/**D07C10/0019c96b6a30ee38 -d "{}"</pre>	
Response	crypto_receipt	JSON object acknowledging receipt of a valid solution.
crypto_receipt	A JSON object containing a preimage that produces a hash with a value less than or equal to that indicated by the difficulty, a timestamp when the signature was produced and a server signature over the preimage concatenated with the timestamp.. An independent auditor who has visibility of this crypto_receipt may validate; 1) the solution, 2) time window, 3) the scavenger server accepted it.	
preimage	String: The preimage data that is passed to the AshMaize hasher.	

timestamp	ISO 8601 date: The date and time when the server signed the receipt
signature	String: A signature produced by the server to signify the solution has been accepted at a specific date and time.The signature is over the preimage concatenated with the timestamp. This signature may be used by an auditor to verify that a solution was both correct and was submitted no later than the timestamp.
Example responses	**When the submitted solution is successful validated** `"crypto_receipt": { "preimage": "0019c96b6a30ee38addr_test1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde8ucclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he**D07C10000FFFFFfd651ac2725e3b9d804cc8b161c0709af14d6264f93e8d4afef0fd1142a3f0112025-10-19T08:59:59.000Z509681483", "timestamp": "2025-10-18T09:01:51.249Z", "signature": "4904f6d82f7075ac21dec9e9c29f6e68d9f36608cc7e7f7912cc62eb4b1b7cf96a83998e8268afb5a6cefedec3331247868d2ae1cee4a736091b3b5acfd25202" }`  **When the submitted address has not been registered** `{ "statusCode":400, "message":"Solution validation failed: Address is not registered" }`  **When the submitted challenge_id is not recognized** `{ "statusCode":404, "message":"Challenge not found: <supplied id>" }`  **When the submitted nonce does not match the required difficulty** `{ "statusCode": 400, "message": "Solution validation failed: Solution does not meet difficulty" }`

Unavailable days	Return on day 0
	`{ "code": "before", "starts_at": "2025-10-30T00:00:00.000Z" }`
	
	Return on day 22
	`{ "code": "after" }`

How to create the nonce?

Creation of the nonce is at the heart of the Scavenger Mine programme. It is created as follows:

1. Initialize the AshMaize ROM using the configuration values defined in the Ashmaize section below and the no_pre_mine value of the specific challenge being solved.
2. Construct a preimage for your registered address by concatenating the following elements:
    a. nonce: 64 bits hex encoded
        - (e.g. 0123456789abcdef)
    b. address: Your registered address
        - (e.g. addr1q8upjxynn626c772r5nzym … c05g3t9xaflw9tmgz6s6m7zhjtjh)
    c. challenge_id: The id of the challenge being solved
        - (e.g. **D21C24)
    d. difficulty: The difficulty setting for the challenge being solved
        - (e.g. 0000FFFF)
    e. no_pre_mine: The no pre-mine setting for the challenge being solved
        - (e.g. 772966a3ff0d3f2b79ff35c718 … 43828b9d226470ac453f0dcb668)
    f. latest_submission: The latest submission time setting for the challenge
        - (e.g. 2025-10-30T23:59:59Z)
    g. no_pre_mine_hour: The hourly no pre-mine setting for the challenge
        - (e.g. 123456789)
3. Hash the preimage and compare the zero bits in the left-most 4 bytes of the hash with the zero bits of the difficulty from the chosen challenge.
4. If the zero bits do NOT match, change the nonce in some way and re-evaluate the AshMaize hash. *The nonce is the only element of the preimage that may be changed for a given challenge/address.*
5. Repeat steps 3 & 4 until the zero bits of the hash match the zero bits of the difficulty.
6. Submit the solution quoting the address, challenge_id and nonce.

POST /donate_to - Re-assign the solutions found by one address to another address

Endpoint POST	/donate_to/{destination_address}/{original_address}/{signature}	
Public access	Yes	
Authentication	No	
Availability days	1-21,22	
Parameters	destination_address	String: A previously registered Cardano address that will ultimately receive the NIGHT tokens.
	original_address	String: A previously registered Cardano address that will donate its tokens to the destination address. It is this address that creates the signature.
	signature	String: A CIP 8/30 compatible signature over a formatted message. The message: “Assign accumulated Scavenger rights to: ” <destination address>
Notes	In order to validate the signature, the Scavenger server will use the public key of original_address that was recorded during POST /register	
Curl	curl -L -X POST https://scavenger.prod.gd.midnighttge.io/donate_to/addr1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde8ucclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he/addr1qrv3cp0m9u7y0elmk0r9wa6an5vfm24ydp5rlau99jxwvaxvj8fgfutr2sevrpsnkx2t6xgqmvdlz8jth8a5phq2wrdqklv2tz/845882a3012704583900d91c05fb2f3c47e7fbb3c657775d9d189daaa468683ff7852c8ce674cc91d284f1635432c18613b194bd1900db1bf11e4bb9fb40dc0a70da6761646472657373583900d91c05fb2f3c47e7fbb3c657775d9d189daaa468683ff7852c8ce674cc91d284f1635432c18613b194bd1900db1bf11e4bb9fb40dc0a70daa166686173686564f4589441737369676e20616363756d756c6174656420536361766656e6765722072696768747320746f3a20616464725f746573743171713464 6c336e68723061787572676372707 56e39787970303470643272326477753578376565616d393870737636646878 6c6465387563636c7632703436686d303737647334767a656c66353536356667336b7937393475687271357570306865584 0514b869f06b1d86b61781291bef81753b118cf9f11a13ee4824ff151cda4529c1580ab465c1aae5153bc18d94de71978221e6d714dd2329634e5733946c39103 -d "{}"	

Example successful response s	**When donation is successful and the original address does not yet have any solutions.**
	{
	"status": "success",
	"message": "Successfully assigned accumulated Scavenger rights from addr1q... to addr1q...",
	"donation_id": "123e4567-e89b-12d3-a456-426614174000",
	"original_address": "addr1q...",
	"destination_address": "addr1q...",
	"timestamp": "2025-08-15T10:30:00.000Z",
	"solutions_consolidated": 0
	}
	
	When assigning for the first time with the destination address having at least one solutions in place
	{
	"status":"success",
	"message":"Successfully assigned accumulated Scavenger rights from addr_test1qrv3cp0m9u7y0 … lz8jth8a5phq2wrdqklv2tz to addr_test1qq4dl3nhr … 6hm077ds4vzelf5565fg3ky794uhrq5up0he",
	"donation_id":"48a9e58e-139b-4a74-94dd-a37096bd35da",
	"original_address":"addr_test1qrv3cp0 … h8a5phq2wrdqklv2tz",
	"destination_address":"addr_test1qq4dl3nhr … g3ky794uhrq5up0he",
	"timestamp":"2025-10-16T14:47:33.925Z",
	"Solutions_consolidated":2
	}
	
	When attempting to donate to self (i.e. undo assignment)
	{
	"status":"success","message":"Successfully undid donation assignment for addr_test1qrv3cp0m9u7y0 … a5phq2wrdqklv2tz",
	"Original_address":"addr_test1qrv3cp0 … jth8a5phq2wrdqklv2tz",
	"Destination_address":"addr_test1qrv3c … jth8a5phq2wrdqklv2tz",
	"timestamp":"2025-10-16T14:11:02.200Z",
	"Solutions_consolidated":0
	}

<table>
<tr>
<td>Example failure responses</td>
<td>

When attempting to assign to the same address more than once


```
{
  "message":"Original address
addr_test1qrv3cp0m9u7y0elmk0r9wa6an5vfm24ydp5rlau99jxwvaxvj8fgfutr
2sevrpsnkx2t6xgqmvdlz8jth8a5phq2wrdqklv2tz already has an active
donation assignment to
addr_test1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde8uc
clv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he",
  "error":"Conflict",
  "statusCode":409
}
```


When attempting to donate to a new address but using an incorrect signature


```
{
  "message":"Invalid CIP-30 signature failed: Message in signature
does not match provided message. Expected signature over message:
\"Assign accumulated Scavenger rights to:
addr_test1qpxvug56xgecxhuzv3c60u4hjkdg6hchqmuevucru7mlvuwy6sd0jxjf
7km7mdk635l26qvu6kwkrsjt8r23r4408jgsxfh364\"",
  "error":"Bad Request",
  "statusCode":400
}
```


When assigning to destination address using an unregistered original address


```
{
  "message":"Original address addr1qrv3cp0m9u … h8a5phq2wrdqklv2tz
is not registered",
  "error":"Not Found",
  "statusCode":404
}
```


</td>
</tr>
<tr>
<td>Unavailable days</td>
<td>

Returned on day 0


```
{
  "code": "before",
  "starts_at": "2025-10-30T00:00:00.000Z"
}
```


</td>
</tr>
</table>

How it works and why it is useful

Participants can utilize multiple Destination addresses to solve challenges, either through separate devices or by using orchestration software on a single machine. The POST /donate_to endpoint allows for the consolidation of solutions from these various addresses into a single smart contract instance. This smart contract will hold the participant's frozen NIGHT, offering two main benefits: simplified NIGHT management from a single wallet, and reduced costs due to lower transaction fees and minimum ADA requirements. Consolidation may occur at any time during mining and up to 24 hours after mining has closed. Consolidation will apply to all claims of the original_address both past and future.

For instance, if a participant is allocated 500 NIGHT, distributed equally across five different participating addresses, and does not use the /donate_to endpoint, they would need to visit the Glacier Drop portal during the Redemption period at least five times, creating five separate redemption smart contract instances on the Cardano mainnet. Given that NIGHT thaws four times during the year, this could amount to up to 20 visits (if the participant chooses to redeem each partial allocation at each thaw event).

However, by using /donate_to to consolidate four of these addresses into one, the participant would only need to visit the Glacier Drop website once (or at most, once per thaw event) and incur only one fee and one minimum ADA. Therefore, at the end of the final thaw period, the single wallet would contain 500 NIGHT.

Note: Only simple (one level) donating from one address to another is allowed. Any attempt to create chains of addresses will be rejected.

GET /work_to_star_rate - Utility function to allow participants to know how many STAR units are allotted to each successful solution

Endpoint GET	/work_to_star_rate
Public access	Yes
Authentication	No
Availability days	0,1-21,22 On days 0 & 1, returns an empty array []
Response	An array of the daily allotment of Star for each crypto_receipt issued.
Curl	`curl https://scavenger.prod.gd.midnighttge.io/work_to_star_rate`

Example response when queried on day 4	`[10882519,7692307,12487254]`

Description	The smallest token unit on the Midnight blockchain is a STAR and there are 1 million STAR per NIGHT.
	
	This endpoint returns the number of STAR allotted to each crypto_receipt issued each day.
	
	To determine the STAR allocation for any given address, simply multiply the number of crypto_receipts issued to that address each day and multiply it by the corresponding number returned in the above array.
	
	Example:
	Suppose an address 'A' successfully solves 24 challenges on day 1 & 2 and 20 challenges on day 3.
	
	`Day 1 STAR allotment is - 24 * 10882519 = 261180456`
	`Day 2 STAR allotment is - 24 *  7692307 = 184615368`
	`Day 3 STAR allotment is - 20 * 12487254 = 249745080`
	
	`Total allotment over the first 3 days is STAR 695540904`
	`i.e NIGHT 695.540904`

AshMaize

AshMaize is an ASIC-resistant variant of the RandomX hash algorithm. It is used with Scavenger Mine to give as many ordinary people as possible the ability to claim NIGHT tokens without the need for specialist computing hardware. You can find the source code for Ashmaize in this repo.

The AshMaize algorithm needs to be configured in a particular manner in order to be compatible with the Scavenger Mine server. This section described the critical aspects of using AshMaize.

Configuring AshMaize	Use the following configuration values for AshMaize.

Field	Value
nbLoops	8
nbInstrs	256

<table>
<tr><td></td><td>
<table>
<tr><td>pre_size</td><td>16777216</td></tr>
<tr><td>mixing_numbers</td><td>4</td></tr>
<tr><td>rom_size</td><td>1073741824</td></tr>
</table>
</td></tr>
<tr><td>AshMaize ROM initialisation</td><td>Whenever AshMaize is used to produce hashes, it must first be initialized using no_pre_mine which will change each day.</td></tr>
<tr><td>Given certain challenge data returned from GET /challenge we may construct an example of using AshMaize</td><td>

```
{
  "challenge_id": "**D07C10",
  "latest_submission": "2025-10-19T08:59:59.000Z",
  "difficulty": "000FFFFF",
  "no_pre_mine":
"fd651ac2725e3b9d804cc8b161c0709af14d6264f93e8d4afef0fd1142a3f0
11",
  "no_pre_mine_hour": "509681483"
}
```

</td></tr>
<tr><td>AshMaize test vector</td><td>Using the challenge data above:

The AshMaize ROM is initialized with:
fd651ac2725e3b9d804cc8b161c0709af14d6264f93e8d4afef0fd1142a3f01
1

The difficulty is:
000FFFFF

For an address previously registered by the participant:
addr_test1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7eeam98psv6dhxlde
8ucclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he

One possible preimage is:
0019c96b6a30ee38addr_test1qq4dl3nhr0axurgcrpun9xyp04pd2r2dwu5x7
eeam98psv6dhxlde8ucclv2p46hm077ds4vzelf5565fg3ky794uhrq5up0he**
D07C10000FFFFFfd651ac2725e3b9d804cc8b161c0709af14d6264f93e8d4af
ef0fd1142a3f0112025-10-19T08:59:59.000Z509681483
Note: Bold is used above for your convenience.

The AshMaize hash is:
000694200fb04137812fb7f35fab2f0e07adf8465397d268bcd97d2f4c7b875
fe6d42f12f377b5b83bcfbd70d6ba55441650c37b8fc80851216b3a1aed7e23
c8

Note: The zero bits in the difficulty are the same as the zero bits of the 4 left most bytes of the hash.

000FFFFF
00069420</td></tr>
</table>