|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document identification number: | | | Classification: open to internal public | |
| Revision date：2025-06-13 | | |  | |
|  | | |  | |
|  | | | | |
| **StorSwift PowerVoting -**  **Filecoin Calibration-CLI test report** | | | | |
|  | | | | |
|  |  | | |  |
|  |  |
|  |  |
|  |  |
|  |  |  | |  |
| StorSwift | | | | |
| June 13, 2025 | | | | |

**Table of Contents**

[Chapter 1 Introduction to Test Report 3](#_Toc7579)

[1 Test purpose 3](#_Toc21823)

[2 Test time/place/personnel 3](#_Toc24584)

[Chapter 2 Test Report 3](#_Toc3190)

[1 Test environment 3](#_Toc6490)

[1.1 Operating system environment 3](#_Toc4311)

[1.2 Hardware environment 3](#_Toc4568)

[2 Module functions 4](#_Toc21145)

[2.1 Test content 4](#_Toc5750)

[2.2 Test cases 4](#_Toc1074)

[3 Test summary 1](#_Toc27639)17

[3.1 Test analysis 1](#_Toc19183)17

[3.2 Test conclusion 1](#_Toc12195)18

**Document change history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Modify content description** | **Modifier** | **Date** | **Remark** |
| 0.0.1 | Full functional test | Cynthia | 2024.06.13 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Chapter 1 Introduction to Test Report

# 1 Test purpose

# To perform comprehensive functional testing of the CLI to confirm whether all features operate normally and meet the release standards.

# 2 Test time/place/personnel

1、Test location: North Financial Port (Wuhan R&D Center)

2、Testing time: April 15, 2025 to June 11, 2025

3、Tester：Cynthia、Yuki Xu

# Chapter 2 Test Report

# 1 Test environment

## Operating system environment

Server operating system: Ubuntu 20.04.5 LTS

Client operating system: Win11 (Google Chrome)

## Hardware environment

The hardware environment is as follows:

Table: Hardware environment

|  |  |  |  |
| --- | --- | --- | --- |
| **Device name** | **Quantity** | **Configuration** | **Equipment usage and description** |
| AMD EPYC 7532 32-Core Processor | 1 | 96 core  1T memory  15T hard disk space | Build PowerVoting server |
| Lenovo Xiaoxin pro | 1 | AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz | Google Chrome visits web pages |

# Module functions

## Test content

PowerVoting - Filecoin Calibration-CLI

## Test cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Serial number | Functional module | Test case name | Case description | Expected results | Pass or not | Tester |
|  | The CLI-proposal list is displayed | List all proposals | When there are multiple proposals in the system, execute the command:. /fil-vote proposal ls to verify that multiple proposals are correctly displayed, including Proposal ID, Creator, Title, and Status; at the same time, verify that the proposal list supports pagination | Implementation: After executing the /fil-vote proposal ls command:   1. The system should correctly display multiple proposals, including the Proposal ID, Creator, Title, and Status of each proposal; 2. At the same time, the proposal list should be displayed on a page and support pagination function to ensure that the proposal list can be displayed in pages when there are too many proposals. |  |  |
|  | The CLI-proposal list is displayed | Data accuracy | 1. Proposal ID: Ensure that the Proposal ID of each proposal is consistent with the proposal ID actually stored in the system; 2. Creator: Verify that the address of the proposal creator (Creator) is correctly displayed and that the format is correct; 3. Title: Ensure that the title (Title) of each proposal is accurately displayed in the output and consistent with the data returned by the database or interface; 4. Status: Verify that the status (Status) of the proposal is correctly displayed and that the status is consistent with the proposal status in the system;   Compare the output data of each field (Proposal ID, Creator, Title and Status) with the data returned by the interface to ensure that each value of the output is consistent with the actual value stored;  Verify that all information is correctly matched when the system returns multiple proposals, and that there are no duplicate, missing, or incorrect data. | 1. Proposal ID: The Proposal ID of each proposal is consistent with the proposal ID actually stored in the system; 2. Creator: The address of the proposal creator (Creator) is correctly displayed and in the correct format; 3. Title: The title (Title) of each proposal is accurately displayed in the output and consistent with the data returned by the database or interface; 4. Status: The status of the proposal (Status) is correctly displayed and consistent with the status of the proposal in the system;   Compare the output data of each field (Proposal ID, Creator, Title and Status) with the data returned by the interface, and each output value is consistent with the actual value stored;  When the system returns multiple proposals, all information is correctly matched and there are no duplicate, missing, or incorrect data. |  |  |
|  | The CLI-proposal list is displayed | The proposal list is empty | When there is no proposal in the system, execute the command:. /fil-vote proposal ls to verify whether the prompt message such as "No Proposal yet!" is displayed and there are no errors or exceptions | Execution command:. /fil-vote proposal ls should display a prompt message, such as "No Proposal Yet!" without errors or exceptions |  |  |
|  | The CLI-proposal list is displayed | The title of the proposal is too long | When the title of the proposal is too long, execute the command:. /fil-vote proposal ls to verify that the system can correctly display the title of the proposal and ensure that it does not cause the interface to crash or the content to be displayed incompletely | 1. When the title of a proposal is too long, the system should be able to display it correctly and completely without affecting the display of other proposals; 2. Other contents in the proposal list (title, status, etc.) should be displayed normally without misalignment or abnormal display; 3. The interface will not crash or lag, and the system should remain stable. |  |  |
|  | The CLI-proposal list is displayed | The title of the proposal contains special characters | When the title of a proposal contains special characters (such as @, #,%, &, \*,!), execute:./fil-vote proposal ls to verify that the system can correctly display the title without affecting other contents in the proposal list | 1. Special characters in the title of the proposal should be displayed correctly and there should be no garbled or display errors; 2. Special characters in the title will not cause the layout of the interface to be confused or the content of the proposal to be lost; 3. Other proposal contents (title, status, etc.) should be displayed normally without abnormal display or misalignment; 4. The system should run stably and no error or exception occurs when executing commands. |  |  |
|  | The CLI-proposal list is displayed | Rules for sorting the proposal list | Execute the fil-vote proposal ls command to verify that the proposal list is in reverse chronological order (i.e., the most recent proposal is at the top) | 1. Proposals in the proposal list should be arranged in reverse order of their end time, so that proposals with later end times are displayed at the top of the list. 2. Proposals with the same deadline for closure shall be ranked according to the time of submission or other rules (e.g. proposal number, etc.). 3. Each proposal in the list should show a clear end time, and the end time should be consistent with the actual end time of the proposal. 4. The sorting result should not be wrong or confused, and all proposals should be displayed in the expected order without duplication or omission. 5. The system should run stably, no error or exception occurs when executing commands, and the proposal sorting does not affect the normal use of other functions. |  |  |
|  | The CLI-proposal list is displayed | two-page separation function | Execution: Run the fil-vote proposal ls command to verify that the paging function in the proposal list works properly:   1. List items display no more than the number set per page, and provide a prompt at the end of each page to access the content of subsequent pages; 2. Check whether the interface displays pagination information (current page number); 3. When switching between different pages, the content in the proposal list should be updated correctly, and no proposals are lost or displayed repeatedly; 4. The page loading speed should be normal, and there is no obvious delay or lag when switching pages; | 1. The proposals in the list should be correctly paginated, with no more than the set number of proposals per page (10 per page); 2. The interface correctly displays the page information (current page number); 3. The data in the proposal list should be accurate and correct, display the proposals with different pages as expected, and there will be no data confusion or loss when switching pages; 4. There is no obvious delay or lag when switching pages; |  |  |
|  | The CLI-proposal list is displayed | Page function-view the page number of the current page | 1. Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the proposal list page: "You are currently on page 1."; 2. When you enter characters according to the instructions to view the next page, the page number prompted at the bottom of the proposal list page changes accordingly-incrementing; 3. When you enter a character as instructed to view the previous page, the page number displayed at the bottom of the proposal list page changes accordingly-decreasing | 1. Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the proposal list page: "You are currently on page 1."; 2. When you enter characters according to the instructions to view the next page, the page number prompted at the bottom of the proposal list page changes accordingly-incrementing; 3. When you enter a character as instructed to view the previous page, the page number displayed at the bottom of the proposal list page changes accordingly-decreasing |  |  |
|  | The CLI-proposal list is displayed | Page function-view the next page | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  n - Next page  p - Previous page   1. Enter the character 'n' and the system returns the list of proposals for the next page; 2. If the current page is already the last page, enter the character 'n' and the message "End of proposals" will appear at the bottom of the page; | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  n - Next page  p - Previous page   1. Enter the character 'n' and the system returns the list of proposals for the next page; 2. If the current page is already the last page, enter the character 'n' and the message "End of proposals" will appear at the bottom of the page; |  |  |
|  | The CLI-proposal list is displayed | Page function-view the previous page | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  n - Next page  p - Previous page   1. Enter the character 'p' and the system returns the list of proposals on the previous page; 2. If the current page is already the first page, enter the character 'p' and the message "You are already on the first page." will be displayed at the bottom of the page; | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  n - Next page  p - Previous page   1. Enter the character 'p' and the system returns the list of proposals on the previous page; 2. If the current page is already the first page, enter the character 'p' and the message "You are already on the first page." will be displayed at the bottom of the page; |  |  |
|  | The CLI-proposal list is displayed | Exit the proposal list page | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  q - Quit to list  Enter the character "q" as prompted at the bottom to exit the proposal list successfully | Execution. /fil-vote proposal ls After the proposal list page is executed, a message is displayed at the bottom of the page:  Enter a command:  q - Quit to list  Enter the character "q" as prompted at the bottom to exit the proposal list successfully |  |  |
|  | CLI-Proposal details display | View details of the proposal | Execution: After executing the /fil-vote proposal ls command, follow the system prompt "Enter Proposal ID to view details:"  Enter a valid Proposal ID and verify whether the system can correctly display the details page information of the Proposal corresponding to the Proposal ID (including Proposal ID, Creator, Title, Content, etc.) | Execution: After executing the /fil-vote proposal ls command, follow the system prompt "Enter Proposal ID to view details:"  Enter a valid Proposal ID and verify whether the system can correctly display the details page information of the Proposal corresponding to the Proposal ID (including Proposal ID, Creator, Title, Content, etc.) |  |  |
|  | CLI-Proposal details display | Input validation of Proposal ID | 1. Proposal ID does not exist 2. Proposal ID is illegal (empty characters, special characters, excessively long characters, etc.) | The system correctly identifies the above situation and gives a prompt "Invalid Proposal ID. Please try again." |  |  |
|  | CLI-Proposal details display | Exit the proposal details page | 1. Execution: After executing the /fil-vote proposal ls command, enter a valid Proposal ID according to the system prompt "Enter Proposal ID to view details:"; 2. At the bottom of the system's return proposal details information, there is the following prompt: "Enter' q' to return to the proposal list or any other key to continue." 3. Enter the character 'q' to exit the proposal details page successfully | 1. Execution: After executing the /fil-vote proposal ls command, enter a valid Proposal ID according to the system prompt "Enter Proposal ID to view details:"; 2. At the bottom of the system's return proposal details information, there is the following prompt: "Enter' q' to return to the proposal list or any other key to continue." 3. Enter the character 'q' to exit the proposal details page successfully |  |  |
|  | CLI-Proposal details display | data integrity | 1. When the status of the proposal is not Complete (Upcoming/In Progress/Vote Counting), ensure that each field of the proposal detail page: Proposal ID, Creator, Title, Content, Start Time, End Time, Snapshot Block Height, Status is correctly displayed without missing data; 2. When the status of the proposal is Complete, ensure that each field of the proposal details page: Proposal ID, Creator, Title, Content, Start Time, End Time, Snapshot Block Height, Status, Vote Approve and Vote Reject are correctly displayed without missing data; | 1. When the status of the proposal is not Complete (Upcoming/In Progress/Vote Counting), the details page of the proposal: Proposal ID, Creator, Title, Content, Start Time, End Time, Snapshot Block Height, Status Each field is correctly displayed without missing data; 2. When the status of the proposal is Complete, the details page of the proposal: Proposal ID, Creator, Title, Content, Start Time, End Time, Snapshot Block Height, Status, Vote Approve and Vote Reject are correctly displayed for each field without missing data. |  |  |
|  | CLI-Proposal details display | Data accuracy | 1. Proposal ID: Verify that it is consistent with the input Proposal ID; 2. Creator: Verify that the address format is in accordance with the standard and that it is consistent with the record at the time of proposal creation; 3. Title, Content: Verify that the title and content of the proposal are displayed correctly and that the content is consistent with the value returned by the interface; 4. Start Time, End Time: Verify that the time format is in line with expectations (e.g., 2025-05-29 16:20); 5. Snapshot block height: Verify that the snapshot block height of the proposal is displayed in thousandths; 6. Status: Verify that the status field of the proposal is correctly displayed and consistent with the value returned by the interface; 7. Vote Approve and Vote Reject (for the proposal status of Complete): Data format: Display in percentage form with two decimal places; Data source: Consistent with the data returned by the interface. | 1. Proposal ID: consistent with the input Proposal ID; 2. Creator: Consistent with the record when the proposal was created, and the address format conforms to the standard; 3. Title, Content: The title and content of the proposal are displayed correctly, and the content is consistent with the value returned by the interface; 4. Start Time, End Time: The time format is in line with expectations (e.g., 2025-05-29 16:20); 5. Snapshot block height: The snapshot block height of the proposal is displayed in the format of thousandths delimiter; 6. Status: The status field of the proposal is correctly displayed and consistent with the value returned by the interface; 7. Vote Approve and Vote Reject (for the proposal status of Complete): Data format: Display in percentage form with two decimal places; Data source: Consistent with the data returned by the interface. |  |  |
|  | CLI-Proposal details display | The title of the proposal is too long | When the proposal title is too long, execute: /fil-vote proposal ls and follow the system prompt "Enter Proposal ID to view details:  ", Enter a valid Proposal ID to verify that the system can correctly display all information on the proposal details page without causing the page to display abnormally or crash | 1. When the proposal title is too long, the system should display it correctly. The proposal details page form should adapt to the page width to ensure that all descriptive information is visible. 2. Other contents in the proposal details form (such as Proposal ID, Content, etc.) should be displayed normally without misalignment or abnormal display. 3. The system should run stably, no error or exception occurs when executing commands, and the interface does not crash or lag. |  |  |
|  | CLI-Proposal details display | The proposal is too long | When the proposal content is too long, execute: /fil-vote proposal ls command and follow the system prompt "Enter Proposal ID to view details:  ", Enter the Proposal ID to verify that the system can correctly display the proposal details page without causing the page to display abnormally or crash | 1. When the proposal content is too long, the system should display it correctly. The proposal details page form should adapt to the page width to ensure that all descriptive information is visible. 2. Other contents in the proposal list (such as Proposal ID, Title, etc.) should be displayed normally without misalignment or abnormal display. 3. The system should run stably, no error or exception occurs when executing commands, and the interface does not crash or lag. |  |  |
|  | CLI- vote | Voting process verification | Verification is performed during the valid voting period:./fil-vote proposal approve--proposalId <ProposalID> --from <walletAddress>   1. Maximum estimated voting fee: The system should return the information indicating the maximum estimated voting fee, such as "The maximum estimated gas cost for voting is: 50662448" 2. User confirmation of voting: At the same time, the system prompts the user to confirm whether to continue voting, and the return message is as follows: "Do you want to proceed with the vote? Type 'yes' to confirm or 'no' to cancel:" 3. Return message hash: The user enters yes to continue the voting operation, and finally returns the correct message hash (MESSAGE HASH) | Verification is performed during the valid voting period:./fil-vote proposal approve--proposalId <ProposalID> --from <walletAddress> command,   1. Maximum estimated voting fee: The system correctly returns the information about the maximum estimated voting fee, such as "The maximum estimated gas cost for voting is: 50662448" 2. User confirmation of voting: At the same time, the system prompts the user to confirm whether to continue voting. The return message is as follows: "Do you want to proceed with the vote? Type 'yes' to confirm or 'no' to cancel:" 3. Return message hash: The user enters yes to continue the voting operation, and finally returns the correct message hash (MESSAGE HASH) |  |  |
|  | CLI- vote | Data accuracy-MESSAGE HASH | 1. Verification MESSAGE HASH Format: The returned MESSAGE HASH should be a valid transaction hash and conform to the standard hexadecimal string format; 2. Validity verification: Through blockchain browser and other tools, ensure that the returned MESSAGE HASH represents the voting operation of this time, and can successfully query the relevant voting transaction information; 3. Voting operation tracking: The voting operation can be accurately tracked through the MESSAGE HASH to ensure the execution and status update of the voting transaction | 1. MESSAGE HASH format: The returned MESSAGE HASH is a valid transaction hash and conforms to the standard hexadecimal string format; 2. Validity verification: Through blockchain browser and other tools, ensure that the returned MESSAGE HASH represents the voting operation of this time, and can successfully query the relevant voting transaction information; 3. Voting operation tracking: The voting operation can be accurately tracked through the MESSAGE HASH to ensure the execution and status update of the voting transaction |  |  |
|  | CLI- vote | Vote during the valid voting period | Execute the command:. /fil-vote proposal approve--proposalId <ProposalID> --from <walletAddress>, ensure that the system correctly returns the feedback information of the voting operation, input yes to confirm the voting, and the system returns the message hash;   1. Verify whether the vote is successful: query the hash value to ensure that the vote operation has been successfully submitted and the vote record is stored correctly; 2. Verify that the voting operation has been recorded: Query the voting status of the proposal to ensure that the voting operation has been correctly recorded and calculated by the system. | 1. Successful vote: The voting operation has been successfully submitted and the voting record has been correctly stored by tracking the execution and status update of the voting transaction corresponding to the MESSAGE HASH; 2. Successful record of voting operation: The voting operation is correctly recorded and calculated by the system by querying the voting situation of the proposal. |  |  |
|  | CLI- vote | Vote time range validation | 1. Try to vote before the proposal starts; 2. After the end time of the proposal, try to vote.   Ensure that the system can correctly process and reject votes performed during invalid time periods to avoid abnormal voting records or feedback | 1. When the user tries to vote before the start time of the proposal, the system should display an error or prompt message such as "The vote has not started" or "The vote time is not in 'in-progress' "; 2. When the user tries to vote after the end time of the proposal, the system should display an error or prompt message such as "vote has ended" or "vote time is not in 'in-progress' "; 3. During the invalid voting period, the voting operation should be rejected and the system will not record any voting behavior; 4. System stability: The system should be stable and crash-free during any voting attempt |  |  |
|  | CLI- vote | Command format validation-missing the necessary parameter proposalId | When the verify command is executed with no--proposalId parameter provided, does the system correctly return an error message indicating that the user has not provided the required parameter, and ensures that no voting operation has been performed and no voting record has been generated:   1. The system returns an error message indicating that the proposalId parameter is missing: "Error: required flag(s)" proposalId "not set"; 2. The system returns detailed Usage information to guide users on how to use the command correctly; 3. Ensure that no voting operations have been performed on the system and that no voting records or cost calculations have been generated | 1. Error message: The system should return a clear error message indicating that the user is missing the proposalId parameter; 2. Unperformed operation: The system will not perform the voting operation due to the lack of necessary parameters, and no records or fees related to the voting will be generated |  |  |
|  | CLI- vote | Command format validation ——---proposalId parameter is missing a value | Verify that when the user executes the command./fil-vote proposal approve--proposalId,  Whether the system can correctly return the error message, prompting the user to lack the value of the--proposalId parameter: "Error: flag needs an argument: --proposalId", and with the Usage prompt information to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--proposalId parameter is missing; 2. The command is not executed: The system does not execute the voting operation due to the missing parameter value |  |  |
|  | CLI- vote | Command format verification —— --from is not a required parameter | To execute a voting command without the--from parameter:./fil-vote proposal approve--proposalId <proposalID>, the system should use the user's default wallet address to complete the voting operation by default, and the command is executed successfully | 1. Without the from parameter: The system should use the default wallet address to vote and the operation should be successfully executed; 2. Operation feedback consistency: The return results of the voting operation should be consistent, regardless of whether the--from parameter is used or not, including the maximum estimated cost, the prompt message to confirm the vote, and the message hash |  |  |
|  | CLI- vote | Command format validation —— --from parameter missing value | Verify that the user executes the following command: /fil-vote proposal approve--proposalId <proposalID> --from  Whether the system can correctly return the error message, prompting the user to lack the value of the--from parameter: "Error: flag needs an argument: --from", and with the Usage prompt information to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--from parameter is missing; 2. The command is not executed: The system does not execute the voting operation due to the missing parameter value |  |  |
|  | CLI- vote | Command format validation —— Execute a command with complete parameters | Enter the command and specify the correct--from and--proposalId parameters:. /fil-vote proposal approve--proposalId <proposalID> --from <walletAddress> The system should use the provided <walletAddress> address to perform the voting operation, and the command is successfully executed | Enter the command and specify the correct--from and--proposalId parameters. The system should use the provided <walletAddress> address to vote for the specified proposal ID, and the voting operation is successfully executed |  |  |
|  | CLI- vote | Proposal ID verification | 1. Proposal ID is empty (no proposal ID parameter value is provided) 2. Proposal ID does not exist 3. Proposal ID is illegal (empty characters, special characters, excessively long characters, etc.) | 1. Proposal ID is empty: When the user does not provide the proposal ID parameter value, the system should return error information, such as: "Error: flag needs an argument: --proposalId"; 2. Proposal ID does not exist: When the provided proposal ID does not exist in the system, the system should return error information, such as: "Failed to submit vote", "error": "invalid Proposal ID"; 3. Proposal ID is illegal:   When the proposal ID contains empty characters, special characters or too long characters, the system should return an error message;  If the proposal ID is too long, the system should limit the length of the proposal ID and return a message such as "Proposal ID is too long, please use an ID with valid length".  The system should reject all invalid proposal IDs mentioned above to prevent invalid IDs from entering the system processing process |  |  |
|  | CLI- vote | The specified <walletAddress> address is invalid | If the specified <walletAddress> address format is incorrect and does not conform to the address specification (address length does not conform, contains illegal characters), the system should correctly return an error message | The system correctly identifies and handles invalid wallet addresses and returns error messages |  |  |
|  | CLI- vote | The specified <walletAddress> address is not imported into the wallet | 1. When the specified address is not imported into the wallet for voting, the system returns the maximum estimated cost and the prompt message to confirm the vote; 2. When the user enters yes to confirm the vote, the system will try to sign and execute the vote; 3. However, the specified wallet address is not imported, and the system cannot find the private key of the wallet to sign. Therefore, the error message "Failed to submit vote" is returned, and the error message "1: Failed to sign message: key not found for <walletAddress>" is returned However, the specified wallet address is not imported, and the system cannot find the private key of the wallet to sign. Therefore, the error message "Failed to submit vote" is returned, and the error message "1: Failed to sign message: key not found for <walletAddress>" is returned | When the user votes for an address that has not been imported into the wallet, the system returns the maximum estimated fee and a prompt message to confirm the vote. When the user completes the confirmation of the vote, the vote fails and the system returns an error message indicating that the signature failed |  |  |
|  | CLI- vote | User confirms the vote-enter yes to confirm the vote operation | Verify that the user can correctly execute the voting operation by entering 'yes' when prompted to confirm whether to continue voting, and ensure that the system returns the correct feedback | 1. System successfully confirms the vote: after the user enters yes, the system should continue to execute the vote operation and return the message hash of the successful vote; 2. Updated voting record: The user's vote should be successfully recorded and can be verified by checking the voting results |  |  |
|  | CLI- vote | User confirms the vote-enter no to cancel the vote operation | Verify that the user can correctly cancel the voting operation by entering 'no' when prompted to confirm whether to continue voting, and ensure that the system returns the correct feedback | 1. Cancel voting feedback: the system returns a clear message to confirm that the voting operation has been cancelled, such as "vote not confirmed, transaction not sent"; 2. No vote submitted: The system should ensure that no voting operation has been performed and no fees will be deducted; 3. The voting status remains unchanged: When querying the voting results of a proposal, ensure that the voting operation has not been recorded. |  |  |
|  | CLI- vote | User confirms vote-invalid vote confirmed | If the user enters an invalid confirmation (such as "nono"), the system should correctly identify and prompt the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and give the user another confirmation vote prompt | If the user enters an invalid confirmation (such as "nono"), the system correctly identifies and prompts the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and again gives the user a confirmation vote prompt |  |  |
|  | CLI- vote | Multiple user voting tests | When multiple users (i.e., different addresses) vote on the same proposal, whether the system can correctly handle each user's voting behavior and ensure that all voting results are accurately counted | 1. Multiple users vote on the same proposal: each user's vote should be recorded independently to ensure that voting data from different addresses do not overlap or conflict with each other; 2. Polling statistics and final results: Ensure that the system correctly counts the total number of votes for each proposal, and ensure that the voting results are not incorrectly covered or lost; 3. Prevent voting data from being overwritten or lost:   Even if there are multiple user votes, the system should avoid any user's vote data being overwritten by other users' data.  If a user changes his vote after voting, the system should update the user's vote data without affecting the votes of other users. |  |  |
|  | CLI- vote | Double voting | 1. Whether the same address can vote on the same proposal more than once 2. Whether the same address can vote on different proposals | Multiple votes on the same proposal at the same address:   1. Users can vote on the same proposal multiple times (for example, by changing their choice or voting again), but each vote updates the previous vote record; 2. The system shall only take the result of the last vote as valid and count it in the final vote statistics; 3. The system can provide hints or feedback, such as "You have already voted and the results have been updated" or "Your latest vote has been recorded".   Voting on different proposals at the same address:   1. The system shall correctly record the independent voting information of users for each proposal; 2. When users vote, the system should be able to accurately distinguish the vote for each proposal and store it in the corresponding proposal record |  |  |
|  | CLI- vote | The account balance is insufficient to deduct the fee | Verify whether the system can correctly handle and prompt the user to fail the vote when the balance of the user's wallet address is insufficient to pay the voting fee:   1. The system should detect the insufficient wallet balance and reject the user's vote request; 2. Ensure that the system does not deduct any fees when the user's balance is insufficient, and prevent the voting operation from being executed incorrectly; 3. When the vote fails, the system should provide the user with a clear error message. | 1. If the user balance is insufficient to pay the voting fee, the system should automatically reject the user's voting request; 2. After the voting operation is rejected, it should be ensured that there is no error deduction or any other operation, and the user's wallet balance should remain unchanged 3. When the vote fails, the system should return a clear error message. |  |  |
|  | CLI-View details of the proposal vote results | Query the details of the proposal results after the vote count is completed | Execute the command:. /fil-vote proposal result <proposalID> to query the result of the proposal after the completion of the vote. The system can correctly display the detailed data of the voting results of the proposal, including the voting address of each voter, the voting weight corresponding to each role, the proportion of the voting weight and the voting result | 1. Complete display of voting results: the system should return all voters and voting information; 2. The voting statistics should be consistent with the actual voting behavior; 3. The query results provided should not contain any missing or duplicated voting data, and all users who participated in the voting should be correctly counted. |  |  |
|  | CLI-View details of the proposal vote results | Details of the results of the proposal-data integrity | The query results should include all the voters who participated in the vote and their corresponding voting information:   1. VOTER ADDRESS: Display the address of all voters to ensure that no one is left out; 2. SP POWER: display the voting weight of each voter to ensure that no one is left out; 3. Client power: display the voting weight of each voter to ensure that no one is left out; 4. Developers' power: display the voting weight of each voter to ensure that no one is left out; 5. TOKEN POWER: display the voting weight of each voter to ensure that no one is left out; 6. Power percentage: shows the proportion of voting weight of each voter to ensure no omissions; 7. RESULT: Display the result of each voter's vote, either Approved or Rejected | The query results should include all the voters who participated in the vote and their corresponding voting information:   1. VOTER ADDRESS: Display the address of all voters to ensure that no one is left out; 2. SP POWER: display the voting weight of each voter to ensure that no one is left out; 3. Client power: display the voting weight of each voter to ensure that no one is left out; 4. Developers' power: display the voting weight of each voter to ensure that no one is left out; 5. TOKEN POWER: display the voting weight of each voter to ensure that no one is left out; 6. Power percentage: shows the proportion of voting weight of each voter to ensure no omissions; 7. RESULT: Display the result of each voter's vote, either Approved or Rejected |  |  |
|  | CLI-View details of the proposal vote results | Details of the results of the proposal-data accuracy | 1. Voter address: The voting address of each voter shall be consistent with the actual voting address in the proposal, and the address format shall comply with the specification; 2. SP POWER: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of B, and shall be rounded to two decimal places; 3. Client power: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of B, and shall be retained to two decimal places; 4. DEVELOPER POWER: The voting weight of each voter should be consistent with the voting weight stored in the system, in units of tFL, and should be rounded to two decimal places; 5. TOKEN POWER: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of tFL, and shall be retained to two decimal places; 6. Power percentage: The voting weight of each voter as a percentage of the total voting weight is correctly calculated and displayed in percentage format, with two decimal places; 7. RESULT: The voting choices (Approved or Rejected) of each voter should be accurate and consistent with their actual voting results; 8. Data source consistency: Ensure that all returned proposal result data comes from the correct database; 9. Data synchronization: The query results should be consistent with the voting data in the database. | 1. Voter address: The voting address of each voter shall be consistent with the actual voting address in the proposal, and the address format shall comply with the specification; 2. SP POWER: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of B, and shall be rounded to two decimal places; 3. Client power: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of B, and shall be retained to two decimal places; 4. DEVELOPER POWER: The voting weight of each voter should be consistent with the voting weight stored in the system, in units of tFL, and should be rounded to two decimal places; 5. TOKEN POWER: The voting weight of each voter shall be consistent with the voting weight stored in the system, in units of tFL, and shall be retained to two decimal places; 6. Power percentage: The voting weight of each voter as a percentage of the total voting weight is correctly calculated and displayed in percentage format, with two decimal places; 7. RESULT: The voting choices (Approved or Rejected) of each voter should be accurate and consistent with their actual voting results; 8. Data source consistency: Ensure that all returned proposal result data comes from the correct database; 9. Data synchronization: The query results should be consistent with the voting data in the database. |  |  |
|  | CLI-View details of the proposal vote results | Details of the results of the proposal-data consistency | 1. Consistency of total voting weight: the sum of all voters' TOKEN POWER should equal the total voting weight of the proposal; 2. Consistency of percentage of votes: the total POWER PERCENTAGE of all voters should be 100%; 3. Consistency of voting choices: Each voter's vote choice (Approved or Rejected) should be consistent with the actual choice at the time of voting. | 1. Consistency of total voting weight: the sum of all voters' TOKEN POWER should equal the total voting weight of the proposal; 2. Consistency of percentage of votes: the total POWER PERCENTAGE of all voters should be 100%; 3. Consistency of voting choices: Each voter's vote choice (Approved or Rejected) should be consistent with the actual choice at the time of voting. |  |  |
|  | CLI-View details of the proposal vote results | Proposal ID verification | 1. Proposal ID is empty (no proposal ID parameter is provided) 2. Proposal ID does not exist 3. Proposal ID is illegal (empty characters, special characters, excessively long characters, etc.) | 1. Proposal ID is empty: When the user does not provide the proposal ID parameter, the system should return an error message, such as: "msg": "No proposal ID provided"}"; 2. Proposal ID does not exist: When the provided proposalID does not exist in the system, the system should return an error message, such as "msg": "Invalid proposal ID", "proposalID": 250}" 3. Proposal ID is illegal:   When the proposal ID contains empty characters, special characters, or too long characters, the system should return an error message, such as "Proposal ID format is invalid".  If the proposal ID is too long, the system should limit the length of the proposal ID and return a message such as "Proposal ID is too long, out of range".  The system should reject all invalid proposal IDs mentioned above to prevent invalid IDs from entering the system processing process |  |  |
|  | CLI-View details of the proposal vote results | Query detailed data on the voting results of an open proposal | Use four pending status (upcoming, in Progress, and vote counting) proposal ID to query the details of the voting results | The message indicates that the vote for the proposal has not yet ended and the final vote result cannot be obtained. At the same time, the message should give the proposalID and the current status of the proposal, such as "msg": "Proposal is not completed, no voting results available", "proposalID": 215, "status": "Upcoming"} "" |  |  |
|  | CLI-View details of the proposal vote results | Details of the results of the single-vote proposals are shown | In the query of the result of a proposal with a single vote, the verification system can correctly display the voting address, voting weight (retained to two decimal places), weight ratio (100%) and voting result (Approved or Rejected) of the unique voter | The system should correctly display the voting address of the unique voter, the voting weight (rounded to two decimal places), the weight ratio (100%), and the voting result (Approved or Rejected) |  |  |
|  | CLI-View details of the proposal vote results | Details of the zero-vote proposal results are shown below | Query the result of a proposal vote when no votes have been received:   1. The system should prompt "No voting"; 2. The percentage of the voting options (Vote Approve, Vote Reject) on the proposal details page is zero | 1. The system should prompt "No voting"; 2. The percentage of the voting options (Vote Approve, Vote Reject) on the proposal details page is zero |  |  |
|  | CLI-Proposal management | help information | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote proposal-h, the system should return the help information, including: command overview, list of available commands, and flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Approve/reject: Used to vote on a proposal 4. LS: Used to list all proposals in pages 5. Results: Display (vote completed) the details of the proposal and its vote | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote proposal-h, the system should return the help information, including: command overview, list of available commands, and flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Approve/reject: Used to vote on a proposal 4. LS: Used to list all proposals in pages 5. Results: Display (vote completed) the details of the proposal and its vote |  |  |
|  | CLI-Proposal management | Help information-Subcommands | 1. Execution. /fil-vote proposal approve-h, /fil-vote proposal reject-h Expected to return the specific help information of the subcommands approve and reject; 2. Execution. /fil-vote proposal ls-h is expected to return specific help information for the ls subcommand; 3. Execute./fil-vote proposal results-h to expect the specific help information returned by the results subcommand | 1. The system should return specific help information: including a brief description of the command function, the basic usage (Usage) of the display command and the supported flag parameters (Flags): --from, -h, --help and--proposalId descriptions to ensure that the help information is clear and accurate and contains all the parameters required by the command; 2. The system should return specific help information: including a brief description of the command function, the basic usage (Usage) of the display command and the supported flag parameters (Flags): -h, --help to ensure that the help information is clear and accurate and contains all the parameters required by the command; 3. The system should return specific help information: brief description of the command function, display the basic usage of the command and (Usage) supported flag parameters (Flags): -h, --help to ensure that the help information is clear and accurate and contains all the parameters required by the command. |  |  |
|  | CLI-Import the wallet through wallet type + private key | Valid wallet type + valid private key | Use the specified wallet type (secp256k1, bls, delegated) + valid private key,  Execute the command: fil-vote wallet add [walletType] [privateKey]. The system will not show any error message indicating that the wallet is successfully imported. You can view the wallet list by executing the command wallet ls to verify whether the imported wallet is successfully added | 1. The system has no error message; 2. Execute the command wallet ls to view the wallet list. The imported wallet address is displayed successfully, and the wallet address format conforms to the specified type: 3. If the type of secp256k corresponds to F1 address, the prefix of the address is t1; 4. If the bls type corresponds to F3 address, then the address prefix is t3; 5. If the delegated type corresponds to the F4 address, the address prefix is t4 |  |  |
|  | CLI-Import the wallet through wallet type + private key | Invalid wallet type | 1. Using custom types not supported by the system; 2. Valid wallet type with spelling errors or mismatched case | The user enters an invalid wallet type. The system identifies the invalid type and returns an error message to indicate that the wallet type is incorrect |  |  |
|  | CLI-Import the wallet through the wallet type + private key | Add the wallet using an invalid private key format | Use a private key with an incorrect or incomplete format to import the wallet | The system returns an error message indicating that the private key format is invalid or incorrect |  |  |
|  | CLI-Import the wallet through the wallet type + private key | The wallet type parameter is missing | Execution command:./fil-vote wallet add [privateKey] When the wallet type parameter is missing, the system should return an error message indicating that the wallet type parameter is missing and a Usage prompt to help users understand the correct format of the command | The system returns an error message indicating that the wallet type parameter is missing, along with a Usage hint to help users understand the correct format of the command |  |  |
|  | CLI-Import the wallet through the wallet type + private key | The private key parameter is missing | Execution command:. /fil-vote wallet add bls, when the private key parameter is missing, the system should return an error message indicating that the private key parameter is missing, and a Usage prompt information is attached to help users understand the correct format of the command | The system returns an error message indicating that the private key parameter is missing, along with a Usage hint to help users understand the correct format of the command |  |  |
|  | CLI-Import the wallet through wallet type + private key | Scene 1: identical type + identical private key | The user has successfully imported a wallet, and the wallet type and private key used already exist. Verify that the system returns an error message indicating that the wallet already exists when you try to import the same type and private key again | The system returns an error message indicating that "the wallet address already exists and cannot be added repeatedly" |  |  |
|  | CLI-Import the wallet through the wallet type + private key | Scene 2: repeated import of private key: different types + same private key | Verify that the system allows successful import of the same private key into different wallet types, and that the address format matches the wallet type | 1. The system has no error message, indicating that the wallet is successfully imported; 2. When viewing the wallet list, the successfully added wallet address should be displayed according to the specified wallet type, that is, the address prefix matches the wallet type (for example, secp256k1 type corresponds to F1 address, and the address prefix is t1; delegated type corresponds to F4 address, and the address prefix is t4) |  |  |
|  | CLI-Set the default wallet address | Set the imported address as the default wallet address | 1. The user has successfully imported the wallet address B, and the wallet address is visible in the wallet list. Execute the command:. /fil-vote wallet ls to confirm that the address B is in the wallet list; 2. Execute the command:. /fil-vote wallet use [address B]; | The system returns details of the command execution:   1. Wallet address: shows the wallet address involved in the current operation, such as address B; 2. Execution status: Success: indicates that the operation is successful and the system has successfully executed the command; 3. Error message (optional): The operation was successful, and the error message is empty. |  |  |
|  | CLI-Set the default wallet address | Set the imported address as the default wallet address | 1. Execute the command:. /fil-vote wallet ls to confirm that address C (not imported wallet address) is not in the wallet list; 2. Execute the command:. /fil-vote wallet use [address C]; | The system returns details of the command execution:   1. Wallet address: shows the wallet address involved in the current operation, such as address C; 2. Execution status: Failed: indicates that the operation fails and the system returns an error message; 3. Error message (optional): The operation failed, and the error message will describe in detail the cause of the failure. |  |  |
|  | CLI-Set the default wallet address | Use the wrong address format to set the default wallet address | Execution command:. /fil-vote wallet use [address D] (address D is in the wrong format) | The system returns details of the command execution:   1. Wallet address: shows the wallet address involved in the current operation, such as address D; 2. Execution status: Failed: indicates that the operation fails and the system returns an error message; 3. Error message (optional): The operation failed, and the error message will describe in detail the cause of the failure. |  |  |
|  | CLI-Set the default wallet address | Address parameter is missing | Execution command:./fil-vote wallet use. When the address parameter is missing, the system should return an error message indicating that the address parameter is missing and a Usage prompt to help users understand the correct format of the command | The system should return an error message indicating the lack of an address parameter, along with a Usage hint to help the user understand the correct format of the command |  |  |
|  | CLI-Set the default wallet address | Set the default wallet address and vote linkage test | 1. The user has successfully imported the wallet address B and executed the command:. /fil-vote wallet use [address B] to set the wallet address B as the default wallet address; 2. To execute a voting command without the--from parameter:./fil-vote proposal approve--proposalId <proposalID>, the system should default to using address B to complete the voting operation | 1. The command to set the default wallet address is successfully executed. Execute the command:. /fil-vote wallet ls, and the address B corresponds to the Default field with X mark; 2. The voting command without the from parameter is successfully executed, and the system returns the message hash. According to the message hash, the voting address is tracked in the blockchain browser as address B, that is, the system successfully completes the voting operation using address B |  |  |
|  | CLI-View the wallet list | The wallet was empty | The current wallet list is empty. Execute the command:. /fil-vote wallet ls | The system returns the message: "The wallet address list is empty" |  |  |
|  | CLI-View the wallet list | List all wallet addresses and mark the default address of the current wallet | After successfully importing multiple valid wallet addresses, execute the following command:. / fil-vote wallet ls | The system returns the wallet list information:   1. Wallet address (Address): List all imported wallet addresses in rows and columns; 2. Default wallet address mark (Default): indicates whether an address is a default wallet address, and only one wallet address should be marked as X to indicate that the address is the current default wallet address, and other wallet addresses do not have X mark; 3. When the default wallet address is not set in the initial state, after executing the command, any wallet address without the X mark should be displayed. |  |  |
|  | CLI-View the wallet list | Verify wallet list sorting rules | Verify that when the command:. /fil-vote wallet ls is executed to list all wallet addresses, the system displays them in accordance with the correct sorting rules:   1. Classification by wallet type: First, classify according to the type of wallet address (t1——t3——t4); 2. Sorted by dictionary order: Wallet addresses are sorted in dictionary order under each type category; 3. No sorting between categories: only addresses within each type are sorted in dictionary order, not across categories | After the command is executed successfully, when all wallet addresses are listed, the system displays them according to the correct sorting rules |  |  |
|  | CLI-View the wallet list | Address format validation | The imported wallet address should match the imported address type, and the matching rules are as follows:   1. Secc256k1 type wallet address format: the address length is 41 characters, and the prefix of the address is t1; 2. The format of bls type wallet address: the address length is 95 characters, and the prefix of the address is t3; 3. Delegated wallet address format: The address is 41 characters long and the prefix is t4 | The successfully imported wallet address matches the imported address type, and the matching rules are as follows:   1. Secc256k1 type wallet address format: the address length is 41 characters, and the prefix of the address is t1; 2. The format of bls type wallet address: the address length is 95 characters, and the prefix of the address is t3; 3. Delegated wallet address format: The address is 41 characters long and the prefix is t4 |  |  |
|  | CLI-View the wallet list | Set the default wallet address and view the wallet list to test and verify the wallet address switching function | 1. The user has successfully imported the wallet address B, and the wallet address is visible in the wallet list. Execute the command:. /fil-vote wallet ls to confirm that the default wallet address in the current wallet list is A; 2. Execute the command:. /fil-vote wallet use [address B]; 3. To execute the command again:./fil-vote wallet ls to view the default wallet address of the current wallet list | 1. The command is executed successfully, and the default wallet address in the current wallet list is A; 2. The command is executed successfully, and the system returns the status of Success; 3. Execute the command again to view the wallet list. The command is executed successfully, and the X mark switches from address A to address B,   That is, successfully switch the default wallet address: A——>B |  |  |
|  | CLI-Wallet management | help information | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote wallet-h, the system should return the help information, including: command overview, list of available commands, and flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Add: Used to import the wallet. 4. LS: Used to list wallets connected to the Lotus, node 5. Use: Used to set the default wallet | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote wallet-h, the system should return the help information, including: command overview, list of available commands, and flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Add: Used to import the wallet. 4. LS: Used to list wallets connected to the Lotus, node 5. Use: Used to set the default wallet |  |  |
|  | CLI-Wallet management | Help information-Subcommands | 1. Execution. /fil-vote wallet add-h is expected to return the specific help information of the add subcommand; 2. Execute./fil-vote wallet ls-h to expect the specific help information returned by the ls subcommand; 3. Run./fil-vote wallet use-h to expect the return of use, the specific help information for the subcommand | 1. The system should return specific help information: brief description of the command function, display the basic usage (Usage) of the command and supported flag parameters (Flags): -h and--help instructions to ensure that the help information is clear and accurate, and contains all the parameters required by the command; 2. The system should return specific help information: including a brief description of the command function, the basic usage (Usage) of the display command, and the supported flag parameters (Flags): -h, --help, to ensure that the help information is clear and accurate, and includes all the parameters required by the command; 3. The system should return specific help information: brief description of the command function, basic usage (Usage) of the display command and supported flag parameters (Flags): -h, --help to ensure that the help information is clear and accurate and contains all the parameters required by the command. |  |  |
|  | CLI-bind the miner ID | Enter a valid minerId | 1. Bind a legal minerId: Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID 2. Bind multiple valid minerIds at the same time: execute the command: /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID\_1 $ACTOR\_ID\_2 | minerId verification is successful, and the system returns user confirmation information to prompt whether to continue the binding operation |  | Yuki Xu |
|  | CLI-bind the miner ID | --from is missing from non-essential parameters | Execution command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID, the system should bind the minerId to the default wallet address | The system binds the default wallet address to the minerId |  | Yuki Xu |
|  | CLI-bind the miner ID | --from parameter specifies that the <WalletAddress> value is missing | Execution command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID--from  Verify that the system can correctly return an error message indicating that the user is missing the value of the--from parameter: "Error: flag needs an argument: --from", along with a Usage message to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--from parameter is missing a value, along with a Usage hint to help users understand the correct format of the command; 2. The command is not executed: The system will not execute the bind minerId operation due to missing parameter values |  | Yuki Xu |
|  | CLI-bind the miner ID | The specified <WalletAddress> address is the wallet address imported into the wallet list | Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID--from <WalletAddress>, and <WalletAddress> is the wallet address imported into the wallet list. The system should bind the minerId to the specified wallet address | The system binds the minerId to the specified wallet address |  | Yuki Xu |
|  | CLI-bind the miner ID | The specified <WalletAddress> address is invalid | Execution command:./fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID--from <WalletAddress>, and if the address format of <WalletAddress> is incorrect and does not conform to the address specification (address length does not conform, contains illegal characters), the system should correctly return an error message | The system correctly identifies and handles invalid wallet addresses and returns error messages |  | Yuki Xu |
|  | CLI-bind the miner ID | The specified <WalletAddress> address is not imported into the wallet | Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID--from <WalletAddress>, and specify that the <WalletAddress> address is not imported into the wallet | The system returns an error message indicating that the specified <WalletAddress> is not imported into the local wallet, resulting in the inability to find the private key |  | Yuki Xu |
|  | CLI-bind the miner ID | User confirms signature-Enter yes to confirm signature | 1. Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID, where $ACTOR\_ID is a valid minerId; 2. The system returns the user confirmation signature prompt message and asks the user to input yes or no; 3. After the user confirms the signature by entering yes, the system continues to execute the command and successfully binds the minerId | After the user enters yes to confirm the signature, the system continues to execute the command and successfully binds the minerId |  | Yuki Xu |
|  | CLI-bind the miner ID | User confirms signature-enter no to cancel signature | 1. Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID, where $ACTOR\_ID is a valid minerId; 2. The system returns the user confirmation signature prompt message and asks the user to input yes or no; 3. After the user enters "no", the system should stop the mineId binding process and return a prompt message indicating that the user has cancelled the signature operation | After the user enters "no", the system should terminate the mineId binding process and return a prompt message indicating that the user has cancelled the signature operation |  | Yuki Xu |
|  | CLI-bind the miner ID | User confirmation signature-Enter invalid confirmation information | 1. Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID, where $ACTOR\_ID is a valid minerId; 2. The system returns the user confirmation signature prompt message and asks the user to input yes or no; 3. Verify that the user enters an invalid confirmation (such as "nono"). The system should correctly identify and prompt the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and again give the user a confirmation signature prompt | If the user enters an invalid confirmation (such as "nono"), the system should correctly identify and prompt the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and give the user another confirmation signature prompt |  | Yuki Xu |
|  | CLI-bind the miner ID | Enter an illegal minerId | Enter an illegal minerId:   1. minerId that starts with "T0" 2. minerId that starts with "t" 3. Other minerIds that do not start with "t0" 4. The minerId does not exist | minerId verification failed, the system returns an error message: Please check your miner ID! |  | Yuki Xu |
|  | CLI-bind the miner ID | Enter a duplicate minerId | Execute the command:. /fil-vote storage-provider claim-miner-actor-ids $ACTOR\_ID\_1 $ACTOR\_ID\_1 | minerId verification failed, the system returns an error message: Your miner ID is duplicated! |  | Yuki Xu |
|  | CLI-bind the miner ID | The input does not belong to the minerId of the current address | It can be successfully submitted and uploaded to the chain, but the verification failure will not bind the success | It can be successfully submitted and uploaded to the chain, but the verification failure will not bind the success |  | Yuki Xu |
|  | CLI-bind the miner ID | Clear the current bound miner ID and submit it | It can be successfully submitted and uploaded to the chain, and the currently bound miner id is in the unbound state | It can be successfully submitted and uploaded to the chain, and the currently bound miner id is in the unbound state |  | Yuki Xu |
|  | CLI-bind the miner ID | Bound status-voting/tallying | Can normally obtain the bound miner ID:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally, 2. Execute the command:. /fil-vote proposal result <proposalID> to view the counting results, which can normally include the power corresponding to the miner id | Can normally obtain the bound miner ID:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally, 2. Execute the command:. /fil-vote proposal result <proposalID> to view the counting results, which can normally include the power corresponding to the miner id |  | Yuki Xu |
|  | CLI-bind the miner ID | Unbound status-voting/tallying | Can normally obtain the unbound status:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally (excluding the miner id that has been unbound), 2. Execute the command:. /fil-vote proposal results <proposalID> to view the vote count results that do not include the power corresponding to the miner ID | Can normally obtain the unbound status:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally (excluding the miner id that has been unbound), 2. Execute the command:. /fil-vote proposal results <proposalID> to view the vote count results that do not include the power corresponding to the miner ID |  | Yuki Xu |
|  | CLI-bind the miner ID | One miner ID is unbound, and the other IDs are in a bound state-voting/counting | Can normally obtain the bound miner ID:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally (excluding the miner id that has been unbound), 2. Execute the command:. /fil-vote proposal results <proposalID> to see that the tally results correctly contain the bound miner ID | Can normally obtain the bound miner ID:   1. Execute the command:. /fil-vote power ls to view the corresponding power normally (excluding the miner id that has been unbound), 2. Execute the command:. /fil-vote proposal results <proposalID> to check that the tally results correctly contain the bound miner ID |  | Yuki Xu |
|  | CLI-bind the miner ID | help information | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote storage-provider-h, the system should return help information, including: a summary of the command, a list of available commands, and flags (Flags) for the command; 2. Available command list and overview of available command functions: claim-miner-actor-ids: declares that the current wallet address is the miner address of the provided participant ID | 1. The user can view the correct format of the command by using the-h or--help parameter:./fil-vote storage-provider-h, and the system should return help information describing the main function of the command; 2. Available command list and overview of available command functions: claim-miner-actor-ids: declares that the current wallet address is the miner address of the provided participant ID |  | Yuki Xu |
|  | CLI-bind the miner ID | Help information-Subcommands | Execute the command:. /fil-vote storage-provider claim-miner-actor-ids-h | The system should return help information: including a brief description of the command function, the basic usage (Usage) of the display command, and the supported flag parameters (Flags): --from, -h, --help, ensuring that the help information is clear and accurate, and includes all the parameters required by the command |  | Yuki Xu |
|  | CLI-GitHub Delegates | Generating signature —— --githubName missing required parameter | Execute the command:. /fil-vote developer generate-github-proof | If the necessary parameters for executing the command are missing, the system should return an error message with help information for the command |  |  |
|  | CLI-GitHub Delegates | The generated signature ——<GithubName> is empty | Execute the command:. /fil-vote developer generate-github-proof--githubName | 1. Error message: The system should return a clear error message, "Error: flag needs an argument: --githubName", indicating that the--githubName parameter is missing a value, and with Usage prompt information to help users understand the correct format of the command; 2. The command is not executed: The system does not generate a signature because the parameter value is missing |  |  |
|  | CLI-GitHub Delegates | The generated signature ——<GithubName> is not empty and valid | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>, where <GithubName> is not empty and legal | The system successfully generates the signature and returns the signature text content and the GistId prompt message to be pasted |  |  |
|  | CLI-GitHub Delegates | The generated signature ——<GithubName> is not empty and illegal | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>,  And <GithubName> is not empty and illegal (does not conform to GitHub user name rules) | The system returns an error message: "GitHub username does not exist." |  |  |
|  | CLI-GitHub Delegates | Generate signature ——<GithubName> Enter a valid but unregistered Github Username | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>,  And <GithubName> is a valid but unregistered GitHub Username | The system returns an error message: "GitHub username does not exist." |  |  |
|  | CLI-GitHub Delegates | --from is missing from non-essential parameters | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>, where <GithubName> is not empty and legal, and the system should use the default wallet address to generate the signature | The system successfully generates the signature, and the returned signature text content contains the walletAddress field. Its value should be the current default wallet address: Use the command wallet ls to view the current default wallet address and ensure that the value of the walletAddress field is consistent with the default wallet address |  |  |
|  | CLI-GitHub Delegates | --From parameter specifies missing <WalletAddress> value | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName> --from,  Verify that the system correctly returns an error message indicating that the user is missing the value of the--from parameter: "Error: flag needs an argument: --from", along with a Usage message to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--from parameter is missing a value, along with a Usage hint to help users understand the correct format of the command; 2. The command is not executed: The system does not generate a signature because the parameter value is missing |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is the wallet address imported into the wallet list | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName> --from <WalletAddress>, and <WalletAddress> is the wallet address imported into the wallet list. The system should use the wallet address specified in the command to generate the signature | The system successfully generates the signature, and the returned signature text content contains the walletAddress field, whose value should be the wallet address specified by the current command |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is invalid | Execution command:./fil-vote developer generate-github-proof--githubName <GithubName> --from <WalletAddress>. If the specified <WalletAddress> address format is incorrect and does not conform to the address specification (address length does not conform, contains illegal characters), the system should correctly return an error message | The system correctly identifies and processes invalid wallet addresses and returns error messages |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is not imported into the wallet | Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName> --from <WalletAddress>, and specify that the <WalletAddress> address is not imported into the wallet | The system returns an error message indicating that the specified <WalletAddress> is not imported into the local wallet, resulting in the inability to find the private key |  |  |
|  | CLI-GitHub Delegates | Generation of signature format and content correctness verification | 1. Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>, where <GithubName> is a valid GitHub user name; 2. Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName> --from <WalletAddress>, where <GithubName> is a valid GitHub user name, and <WalletAddress> is an address imported into the wallet list   After the verification command is executed, verify that the Signature format and content are correct, and that the githubName is the <GithubName> submitted when the signature was generated; and that the walletAddress is the current default wallet address or the specified address | After executing the command, the Signature returned by the system is in the correct format and content, and the githubName is the <GithubName> submitted when the signature is generated; the walletAddress is the current default wallet address or the specified address |  |  |
|  | CLI-GitHub Delegates | Verify signature according to the specified GistId —— --gistId is missing a necessary parameter | Execute the command:. /fil-vote developer claim-github-proof | If the necessary parameters are missing, the system returns an error message with command help information |  |  |
|  | CLI-GitHub Delegates | Verify the signature according to the specified GistId —— <GistId> is empty | Execute the command:. /fil-vote developer claim-github-proof--gistId | 1. Error message: The system should return a clear error message, "Error: flag needs an argument: --githubName", indicating that the--githubName parameter is missing a value, and with Usage prompt information to help users understand the correct format of the command; 2. The command is not executed: The system does not generate a signature because the parameter value is missing |  |  |
|  | CLI-GitHub Delegates | Verify the signature according to the specified GistId —— <GistId> is not empty and legal | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId> is not null and legal | After the command is executed, the system returns  The total cost rate set by the sender  Prompt the user to confirm whether to continue the operation |  |  |
|  | CLI-GitHub Delegates | The signature is verified according to the specified GistId —— <GistId> is not empty and illegal | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId> is not null and illegal | After the command is executed, the system returns  The total cost rate set by the sender  Prompt the user to confirm whether to continue the operation |  |  |
|  | CLI-GitHub Delegates | --from is missing from non-essential parameters | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId> is not null and legal | After the command is executed, the system returns  The total cost rate set by the sender  As well as a prompt message to prompt the user to confirm whether to continue the operation |  |  |
|  | CLI-GitHub Delegates | --From parameter specifies missing <WalletAddress> value | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId> --from，  Verify that the system correctly returns an error message indicating that the user is missing the value of the--from parameter: "Error: flag needs an argument: --from", along with a Usage message to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--from parameter is missing a value, along with a Usage hint to help users understand the correct format of the command; 2. The command is not executed: The system does not generate a signature because the parameter value is missing |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is the wallet address imported into the wallet list | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId>--from <WalletAddress>, and the specified <WalletAddress> address is an imported wallet address in the wallet list | After the command is executed, the system returns  The sender sets the total cost rate and prompts the user to confirm whether to continue the operation |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is invalid | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId>--From <WalletAddress>, and the specified <WalletAddress> address format is incorrect, does not conform to the address specification (address length does not conform, contains illegal characters), the system should correctly return an error message | The system correctly identifies and processes invalid wallet addresses and returns error messages |  |  |
|  | CLI-GitHub Delegates | The specified <WalletAddress> address is not imported into the wallet | Execute the command:. /fil-vote developer claim-github-proof--gistId  <GistId>--from <WalletAddress>, and the specified <WalletAddress> address is not imported into the wallet | 1. After the command is executed, the system returns the total cost rate set by the sender and prompts the user to confirm whether to continue the operation; 2. After the user enters yes to confirm, the system returns an error message indicating that the specified <WalletAddress> is not imported into the local wallet, resulting in the inability to find the private key |  |  |
|  | CLI-GitHub Delegates | Confirm whether to continue the GitHub proof operation-Enter yes to confirm | 1. Execute the command:. /fil-vote developer claim-github-proof--gistId <GistId>, where <GistId> is valid and legal; 2. The system returns the user confirmation prompt message to continue the operation, and asks the user to input yes or no; 3. The user enters yes to confirm the operation, the system continues to execute the command and verifies the GistId, and the system returns the message hash (MESSAGE HASH) | After the user enters yes to confirm the signature, the system continues to execute the command and successfully authorizes the system and returns the message hash (MESSAGE HASH) |  |  |
|  | CLI-GitHub Delegates | Confirm whether to continue the GitHub proof operation-Enter no to cancel and continue | 1. Execute the command:. /fil-vote developer claim-github-proof--gistId <GistId>, where <GistId> is valid and legal; 2. The system returns the user confirmation prompt message to continue the operation, and asks the user to input yes or no; 3. After the user enters no to cancel, the system will not continue to execute the command and return a prompt message | After the user enters no to cancel the signature, the system will not continue to execute the command and return the prompt message: "UpdateGistId is not confirmed, transaction is not sent." |  |  |
|  | CLI-GitHub Delegates | Confirm whether to continue the GitHub proof operation-Enter invalid confirmation information | 1. Execute the command:. /fil-vote developer claim-github-proof--gistId <GistId>, where <GistId> is valid and legal; 2. The system returns the user confirmation signature prompt message and asks the user to input yes or no; 3. Verify that the user enters an invalid confirmation (such as "nono"). The system should correctly identify and prompt the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and again give the user a confirmation signature prompt | If the user enters an invalid confirmation (such as "nono"), the system should correctly identify and prompt the user to enter a valid confirmation "Invalid input, please type 'yes' or 'no'." and again give the user a prompt to confirm whether to continue the GitHub proof operation |  |  |
|  | CLI-GitHub Delegates | Verify that the check is successful | 1. Execute the command:. /fil-vote developer generate-github-proof--githubName <GithubName>, where Github\_Username is the registered Github name; 2. The system returns the signature text content and the GistId prompt message; 3. Create a Gist using the signature information returned by the system to generate a GistId; 4. Execute the command:. /fil-vote developer claim-github-proof--gistId <GistId>, where <GistId> is valid and legal; 5. After the user inputs yes to confirm, the message hash is returned and the chain is waiting for success; 6. After the successful uploading, GistId is verified successfully and GitHub authorization is completed; 7. To execute the command:. /fil-vote power ls, you can view the power of the developer role at the corresponding address (default address or specified address) normally | The verification is successful and the developer role weight corresponding to the address can be successfully displayed |  |  |
|  | CLI-GitHub Delegates | Verification of failed logic 1-Verification of the consistency between the generated Signature and the GitHub name corresponding to the GitHub ID | The generated Signature corresponds to the consistency check between the GitHub name and the GitHub name used to create the Gist:   1. Do not create gists using the GitHub name included in the signature 2. The parameter <GithubName> is a mixed case of the [GitHub name] used to create a Gist | The verification failed, and the developer role weight corresponding to the address could not be successfully displayed |  |  |
|  | CLI-GitHub Delegates | Verification of failure logic 2-Verify the consistency between the address/githubName in the generated Signature and the GistId | 1. When creating a Gist, the content is any text that does not contain a complete Signature 2. The generated Signature is not related to the GistId (which corresponds to other Signatures) | The verification failed, and the developer role weight corresponding to the address could not be successfully displayed |  |  |
|  | CLI-GitHub Delegates | Repeat authorization test | 1. Address 001 successfully uses github account A for authorization, and address 001 uses github account B for authorization again; 2. Address 001 creates multiple gists (01 and 02) using the same Signature, enters one of the gistids (01), and then uses the gistid (02) to authorize it again | 1. The authorized account at address 001 is normally changed from GitHub account A to GitHub account B; 2. The authorized account of address 001 remains unchanged, and the gistid related information is updated from 01 to 02 |  |  |
|  | CLI-GitHub Delegates | Authorize coverage testing | 1. Address 001 successfully authorized with GitHub account A; 2. Address 002 successfully authorized with GitHub account B; 3. Address 001 uses GitHub account B to authorize again | 1. The authorized account at address 001 is normally changed from GitHub account A to GitHub account B 2. The authorization status of address 002 is cancelled |  |  |
|  | CLI-GitHub Delegates | Bound status-voting/tallying | 1. You can normally access the bound GitHub account 2. Execute the command:. /fil-vote power ls to view the corresponding role's power normally, 3. Execute the command:. /fil-vote proposal result <proposalID> to view the voting results, which can normally include the power corresponding to the github account | 1. You can normally access the bound GitHub account 2. Execute the command:. /fil-vote power ls to view the corresponding role's power normally, 3. Execute the command:. /fil-vote proposal result <proposalID> to view the voting results, which can normally include the power corresponding to the github account |  |  |
|  | CLI-GitHub Delegates | Unbound status-vote/count | 1. Can normally obtain the unbound status 2. Execute the command:. /fil-vote power ls, the corresponding power (excluding the already unbound github account) can be viewed on the voting page normally, 3. Execute the command:. /fil-vote proposal result <proposalID> to view that the vote count does not include power corresponding to the github account | 1. Can normally obtain the unbound status 2. Execute the command:. /fil-vote power ls, the corresponding power (excluding the already unbound github account) can be viewed on the voting page normally, 3. Execute the command:. /fil-vote proposal result <proposalID> to view that the vote count does not include power corresponding to the github account |  |  |
|  | CLI-GitHub Delegates | help information | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote developer-h, the system should return the help information, including: command overview, list of available commands, flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Generate-github-proof: Generate signature 4. Claim-github-proof: Verify that the provided GitsId contains the corresponding signature content | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote developer-h, the system should return the help information, including: command overview, list of available commands, flags (Flags) of the command; 2. List of available commands and overview of available command functions: 3. Generate-github-proof: Generate signature 4. Claim-github-proof: Verify that the provided GitsId contains the corresponding signature content |  |  |
|  | CLI-GitHub Delegates | Help information-Subcommands | 1. Execute the command:. /fil-vote developer generate-github-proof-h 2. Execute the command:. /fil-vote developer claim-github-proof-h | 1. The system should return help information: brief description of the command function, basic usage (Usage) of the display command and supported flag parameters (Flags): --from, --githubName, -h, --help, ensuring that the help information is clear and accurate and contains all the parameters required by the command 2. The system should return help information: brief description of the command function, basic usage (Usage) of the display command, and supported flag parameters (Flags): --from, --gistId, -h, --help, ensuring that the help information is clear and accurate, and contains all the parameters required by the command |  |  |
|  | CLI-View the power of each role | List the voting rights of each role for all addresses in the wallet | Execute the command:. /fil-vote power ls--day <day>, where day is the valid date and conforms to the YYYYMMDD format | For the effective date (in YYYYMMDD format), the system displays the voting rights information of all wallets corresponding to the corresponding date as required: returns the wallet address and the voting rights data of each role corresponding to it |  |  |
|  | CLI-View the power of each role | --Day missing necessary parameters | Execute the command:./fil-vote power ls  Verify that the system returns an error message indicating that the necessary parameter day is missing: "Error: required flag (s)" proposalId "not set";  It also returns detailed Usage information to guide users on how to use the command correctly | When the necessary parameters for executing the command are missing, the system should return an error message indicating that the necessary parameters are missing, along with help information for the command |  |  |
|  | CLI-View the power of each role | --Missing values for day necessary parameters | Execute the command:./fil-vote power ls--day  Verify that the system correctly returns an error message indicating that the user is missing the value of the--day parameter: "Error: flag needs an argument: --day", along with a Usage message to help the user understand the correct format of the command | 1. Error message: The system should return a clear error message indicating that the--day parameter is missing a value, and an Usage message to help users understand the correct format of the command; 2. The command is not executed: The system does not execute the view power operation due to missing parameter values |  |  |
|  | CLI-View the power of each role | day input validation | 1. Valid date format: The date should be in the format of YYYYMMDD; 2. Effective date range: The date should be a real date and not a future date; 3. Boundary case: Ensure that the user enters a valid date, excluding an invalid date such as 2025-02-30 | 1. Valid date format and validity period: the system can correctly identify and continue to execute the command; 2. Illegal date format/future date/non-existent date: The system should return an error message and prompt the user to enter a YYYYMMDD date |  |  |
|  | CLI-View the power of each role | Address information-Data consistency | 1. The wallet address in./fil-vote wallet ls and./fil-vote power ls output is consistent; 2. Each address should be correctly matched to the corresponding Power data 3. The corresponding Power, data (such as SP POWER) for each address should match the actual role of that address | 1. The wallet address in./fil-vote wallet ls and./fil-vote power ls output is consistent; 2. Each address should be correctly matched to the corresponding Power data 3. The corresponding Power, data (such as SP POWER) for each address should match the actual role of that address |  |  |
|  | CLI-View the power of each role | Address information-Address sorting consistency | 1. The sorting of the address list should be consistent in the output of the two commands./fil-vote wallet ls and./fil-vote power ls; 2. Sorting should be done first by wallet type (e.g. t1, t3) and then in alphabetical order | 1. The sorting of the address list should be consistent in the output of the two commands./fil-vote wallet ls and./fil-vote power ls; 2. Sorting should be done first by wallet type (e.g. t1, t3) and then in alphabetical order |  |  |
|  | CLI-View the power of each role | Role voting information-data display | The role voting rights information should be displayed in the order of SP, Client, Developer, and Tokenholder   1. Role: SP, Client, Developer, Tokenholder; 2. Power: The number of voting rights should be retained to two decimal places; if a role has no voting rights, it will be displayed as 0; 3. Voting rights Power unit: Voting rights should be shown according to the actual situation. The unit of SP POWER and CLIENT POWER is B, and the unit of DEVELOPER POWER and TOKENHOLDER POWER is tFIL (or FIL, depending on system Settings) | The role voting rights information should be displayed in the order of SP, Client, Developer, and Tokenholder   1. Role: SP, Client, Developer, Tokenholder; 2. Power: The number of voting rights should be retained to two decimal places; if a role has no voting rights, it will be displayed as 0; 3. Voting power Power unit: The corresponding unit of voting power is shown according to the actual situation |  |  |
|  | CLI-View the power of each role | Role voting information-data integrity | Verify that the system displays the voting rights of the current wallet for all roles (SP, Client, Developer, Tokenholder): voting weight to ensure that no data is omitted for any role | The system correctly displays the voting rights data of all roles. Each role (SP, Client, Developer, Tokenholder) has the corresponding voting rights value, and the data is not missing |  |  |
|  | CLI-View the power of each role | Role voting information-data accuracy | Verify that the role voting rights information displayed by the system is accurate and ensure that the data source is correct and consistent with the data;  Precision: retain two decimal places when displayed, and avoid rounding off the decimal part of the number to ensure that the numerical precision is not lost | The voting power data displayed in the system is consistent with the actual data on the blockchain, the value of the role voting power is accurate, and there is no abnormal data source;  Precision: retain two decimal places when displayed, and avoid rounding off the decimal part of the number to ensure that the numerical precision is not lost |  |  |
|  | CLI-View the power of each role | The wallet address does not have voting rights in any of the roles | 1. The wallet address already exists and does not have voting rights in any role; 2. Check the voting rights of the role: execute the command:./fil-vote power ls--day <day>, where day is the valid date and conforms to the format of YYYYMMDD; 3. Verify that the voting weight of the wallet address under each role should be 0 | Zero voting rights: The system should return zero voting rights for the wallet address on all roles (SP, Client, Developer, Tokenholder) |  |  |
|  | CLI-View the power of each role | Some roles in the wallet address have voting rights | 1. Some roles in the wallet address have voting rights, while others have zero; 2. Execute the command:./fil-vote power ls--day <day>, where day is the valid date and conforms to the format of YYYYMMDD; 3. Verify that the system can correctly display the voting rights data of each role, and the wallet address only shows the data of the roles with voting rights, while other roles without voting rights are displayed as zero, to ensure the accuracy and consistency of the data | The voting rights of all roles are displayed correctly, and the voting rights of each role are consistent with the original data, the unit is correct, and the display format is two decimal places |  |  |
|  | CLI-View the power of each role | help information | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote power-h, the system should return the help information, including: command overview, list of available commands, flags (Flags) of the command; 2. Available command list and overview of available command functions: ls: List the weight of each role on a specified date | 1. The user can view the correct format of the command by using the-h or--help parameter:. /fil-vote power-h, the system should return the help information, including: command overview, list of available commands, and flags (Flags) of the command; 2. Available command list and overview of available command functions: ls: List the weight of each role on a specified date |  |  |
|  | CLI-View the power of each role | Help information-Subcommands | Execute the command:. /fil-vote power ls-h, and the system should return the help information: including a brief description of the function of the command, the basic usage (Usage) of the display command, and the supported flag parameters (Flags): --day, -h, and--help | The system should return help information: including a brief description of the command function, the basic usage (Usage) of the display command, and the supported flag parameters (Flags): --day, -h, --help, ensuring that the help information is clear and accurate, and includes all the parameters required by the command |  |  |

# 3 Test summary

## 3.1 Test analysis

After multiple rounds of testing and regression testing of the PowerVoting - Filecoin Calibration-CLI , the inherited bugs are currently within controllable range. In the functional test, we covered all the main functional modules of the software, including proposal list display, proposal details, voting, viewing proposal voting result details, wallet (type + private key), display default wallet address, wallet list, binding miner id, GitHub authorization and most of the functional modules corresponding to each role performed well and met the expected functional requirements. The reliability test mainly evaluates the stability and robustness of the software under long-term operation and abnormal conditions. The software performed stably in the long-term running test, and no serious crashes or abnormalities occurred.

## 3.2 Test conclusion

After comprehensive testing of the PowerVoting - Filecoin Calibration CLI command-line interface for functionality, performance, and stability, the following conclusions have been drawn:

1. Functionality Testing: All system functions operate as expected, including proposal list display, proposal detail viewing, voting, viewing detailed voting results, wallet import (type + private key), setting default wallet address, viewing the wallet list, binding miner ID, GitHub authorization, and viewing power by role. These features meet design specifications and user requirements.
2. Performance Testing: Under normal usage conditions, the system responds quickly, all operations are smooth, and there is no noticeable lag or delay. Overall performance is excellent.
3. Stability Testing: The system underwent long-term operation testing without experiencing crashes, data loss, or other critical issues, demonstrating high stability.
4. Security Testing: The system's user authentication and access control functions work effectively, ensuring the security and confidentiality of system data.

In summary, the PowerVoting - CLI performed well during testing, demonstrating high stability, excellent performance, and comprehensive functionality, making it suitable for practical use.