

Iroha 2 requirements

General info

| | |
|---------------------|--|
| Target release | Iroha 2.0 MVP |
| Document status | DRAFT |
| Document owner | Vadim Reutskiy |
| Tech lead | TBD |
| Developers | Nikita Puzankov Egor Ivkov Mikhail Boldyrev |
| Blockchain advisors | Salakhiev Kamil, Andrei Lebedev, Bulat Nasrullin, |
| Stakeholders | Iurii Vinogradov, Makoto Takemiya, Zilya Yagafarova, Pavel Golovkin, Anton Khvorov |
| QA | Stephan Lavrentev |

Goals

- TBD

Background and strategic fit

TBD

Sources

- Iroha 2 white-paper: https://github.com/hyperledger/iroha/blob/iroha2-wp/iroha_2_whitepaper.md
- Sora 2 project requirements
 - Anton Khvorov
 - Zilya Yagafarova
 - Pavel Golovkin
 - Bogdan Mingela
 - Ruslan Rezin
- Bakong project requirements
 - Zilya Yagafarova
 - Andrey Marin
- KYC project requirements
- Other internal project requirements
 - Iurii Vinogradov

Assumptions

- TBD

Table of contents

- General info
- Goals
- Background and strategic fit
- Sources
- Assumptions
- Table of contents
- Changes history
- Requirements
 - Actors
 - Functional requirements
 - [FR0000] Example use-case; ID should be unique
 - 00. Iroha network operations
 - [FR0001] Starting the Iroha network
 - [FR0002] Adding peer to the Iroha network
 - [FR0003] Removing peer from the Iroha network
 - [FR0004] Configuring initial state of the Iroha network
 - [FR0005] Changing configuration of working Iroha network
 - [FR0006] Changing configuration of the particular peer
 - 01. Making changes in Iroha network data by Iroha special instructions
 - [FR0100] Sending the transaction to the Iroha network
 - [FR0101] Creation of the account in the Iroha network
 - [FR0102] Configuring permissions for the account in the Iroha network
 - [FR0102] Granting permissions for the account in the Iroha network
 - [FR0104] Sending complex instruction using ISI DSL
 - [FR0105] Sending instruction and subscribing to the status of finalization
 - [FR0106] Creation of the multi-signature account in the Iroha network
 - [FR0107] Changing quorum for the multi-signature account
 - [FR0108] Changing list of signatories for the multi-signature account
 - [FR0109] Signing multi-signature transaction
 - [FR0110] Changing the conditions for the multi-signature account
 - [FR0111] Assigning weights to the signatories of the multi-signature account
 - [FR0112] Associating and changing arbitrary data payload with the account
 - [FR0113] Sending instruction with the payload
 - [FR0114] Sending non-fungible assets
 - 02. Acquiring data from the Iroha network by queries
 - [FR0200] Acquiring data from the Iroha network by query
 - [FR0201] Acquiring the information about the selected account
 - [FR0202] Acquiring of the current permissions for the selected account
 - [FR0203] Acquiring a list of pending multi-signature transactions

- [FR0204] Acquiring a list of current conditions for a multi-signature account
- [FR0205] Acquiring a block by its number
- [FR0206] Acquiring blocks subscription
- [FR0207] Acquiring pending transactions subscription
- [FR0208] Subscribing on the query results
- [FR0209] Validate result of the query
- [FR0210] Query old Iroha state (e. g., query balance month ago)
- [FR0211] Querying list of accounts with the predefined filter
- [FR0212] Retrieving the list of keys of data payload, associated with the target account
- [FR0213] Retrieving the value of data payload by the key, associated with the target account
- [FR0214] Querying list of transactions with predefined filter
- [FR0215] Subscription on incoming transactions
- [FR0216] Requesting list of non-fungible assets in account
- [FR0217] Requesting data from the block storage by using special request language
- 03. Setting up and executing triggers
 - [FR0300] Setting up a trigger in the Iroha network
 - [FR0301] Manually firing the trigger
 - [FR0302] Removing the trigger
- 09. High-level use cases
 - [FR0900] Configuration of fees for transfers inside the current Iroha network
 - [FR0901] Sending transaction with fees
 - [FR0902] Delegation of account control with time limit
 - [FR0903] Inheritance of the account after period of inactivity
 - [FR0904] Distribution of fees according to business rules of the project
 - [FR0905] Managing the list of signatories
 - [FR0906] Making decisions by parliament voting
 - [FR0907] Management of non-fungible assets
 - [FR0908] Nominating validator by staking assets
 - [FR0909] Slashing of stakes for inappropriate behaviour of validator
 - [FR0910] Obtaining a list of trusted peers
 - [FR0911] Configuring a minimum limit of assets needed to create an account
 - [FR0912] Creation of account with configured minimum amount of tokens
- Non-functional requirements
 - [NFR0000] Example quality attribute; ID should be unique
 - 00. Performance
 - [NFR0001] Transaction processing speed
 - [NFR0001] Delay of block creation
 - [NFR0002] Delay of restarting the peer
 - [NFR0003] Performing as expected on predefined hardware

- [NFR0004] Providing enough capacity for user's accounts
- 01. Portability
 - [NFR0100] Easy integration from client side applications
 - [NFR0101] Horizontal scalability of the network size
 - [NFR0102] Adaptability for different environments and projects
 - [NFR0102] Flexibility of integrated DSL for complex operations and triggers
 - [NFR0103] Reusability of the Iroha interface during integration with external systems
 - [NFR0103] Configurability of permission
- 02. Security
 - [NFR0200] Non-repudiation of data between peer and client
- 03. Usability
 - [NFR0300] Convenient documentation for different user types
- 04. Reliability
 - [NFR0400] Available proofs of efficiency of technical decisions and implementation
 - [NFR0401] Safety of the integrated DSL language for triggers

- [Questions](#)
- [Not Doing](#)

Changes history

| Version | Published | Changed By | Comment |
|-------------------------|---------------------------|-----------------------|--|
| CURRENT (v. 100) | Oct 07, 2020 09:03 | Vadim Reutskiy | |
| v. 100 | Sep 23, 2020 05:39 | Vadim Reutskiy | |
| v. 99 | Sep 23, 2020 05:37 | Vadim Reutskiy | Reviewed and fixed some FRs, added links to related FRs. |
| v. 98 | Sep 23, 2020 04:54 | Vadim Reutskiy | |
| v. 97 | Sep 23, 2020 04:51 | Vadim Reutskiy | |
| v. 96 | Sep 19, 2020 12:07 | Vadim Reutskiy | |
| v. 95 | Sep 19, 2020 11:55 | Vadim Reutskiy | Added NFR0103 about flexibility of permissions |
| v. 94 | Sep 19, 2020 11:30 | Vadim Reutskiy | |
| v. 93 | Sep 19, 2020 11:28 | Vadim Reutskiy | |
| v. 92 | Sep 19, 2020 11:21 | Vadim Reutskiy | |
| v. 91 | Sep 19, 2020 11:20 | Vadim Reutskiy | Added NFR0103 about convenient API |
| v. 90 | Sep 19, 2020 09:44 | Vadim Reutskiy | Added FR0217 with custom query language |

| | | | |
|-------|--------------------|--------------------------------|---|
| v. 89 | Sep 19, 2020 09:43 | Vadim Reutskiy | |
| v. 88 | Sep 19, 2020 09:35 | Vadim Reutskiy | |
| v. 87 | Sep 19, 2020 09:25 | Vadim Reutskiy | Added NFR0400 with requirement to verifiable and repeatable documentation about decisions |
| v. 86 | Sep 19, 2020 09:18 | Vadim Reutskiy | Added NFR0300 about documentation |
| v. 85 | Sep 19, 2020 08:45 | Vadim Reutskiy | Added NFD0004 about network capacity for accounts |
| v. 84 | Sep 19, 2020 08:44 | Vadim Reutskiy | |
| v. 83 | Sep 19, 2020 08:30 | Vadim Reutskiy | Filled FR0902 about access delegation |
| v. 82 | Sep 19, 2020 08:19 | Vadim Reutskiy | |
| v. 81 | Sep 19, 2020 08:09 | Vadim Reutskiy | Added FR0114 and FR0216 about non-fungible assets |
| v. 80 | Sep 19, 2020 07:08 | Vadim Reutskiy | Added FR0114 about operations with non-fungible assets |
| v. 79 | Sep 19, 2020 06:50 | Vadim Reutskiy | |
| v. 78 | Sep 16, 2020 10:29 | Vadim Reutskiy | |
| v. 77 | Sep 16, 2020 10:08 | Vadim Reutskiy | |
| v. 76 | Sep 16, 2020 10:06 | Vadim Reutskiy | |
| v. 75 | Sep 16, 2020 09:52 | Vadim Reutskiy | |
| v. 74 | Sep 11, 2020 11:29 | Vadim Reutskiy | |
| v. 73 | Sep 11, 2020 11:22 | Vadim Reutskiy | Add FR0907 with non-fungible tokens |
| v. 72 | Sep 11, 2020 08:51 | Vadim Reutskiy | Update FR0300: add note about reusing the same state. |
| v. 71 | Sep 11, 2020 08:46 | Vadim Reutskiy | Update FR0904 according to comments by Iurii |
| v. 70 | Sep 11, 2020 08:28 | Vadim Reutskiy | |
| v. 69 | Sep 11, 2020 07:17 | Vadim Reutskiy | Add NFR0102 about DSL flexibility |
| v. 68 | Sep 11, 2020 06:53 | Vadim Reutskiy | Added FR0906 for parliament voting |
| v. 67 | Sep 11, 2020 05:50 | Vadim Reutskiy | |
| v. 66 | Sep 11, 2020 05:47 | Vadim Reutskiy | Added FR0905 with management of MST signatories |

| | | | |
|-------|--------------------|--------------------------------|--|
| v. 65 | Sep 10, 2020 11:52 | Vadim Reutskiy | |
| v. 64 | Sep 10, 2020 11:30 | Vadim Reutskiy | |
| v. 63 | Sep 10, 2020 11:05 | Vadim Reutskiy | Added NFR0102 with requirements to system performance flexibility |
| v. 62 | Sep 10, 2020 10:59 | Vadim Reutskiy | Added performance requirements from Sora 2 |
| v. 61 | Sep 10, 2020 10:57 | Vadim Reutskiy | Added NFR0003 with requirements to the minimal machine configuration |
| v. 60 | Sep 10, 2020 07:22 | Vadim Reutskiy | Added NFR0101 for network scalability |
| v. 59 | Sep 10, 2020 07:02 | Vadim Reutskiy | Added security requirements NFR0200 |
| v. 58 | Sep 10, 2020 05:46 | Vadim Reutskiy | Added quality attribute for SDK availability for platforms (NFR0100) |
| v. 57 | Sep 09, 2020 05:36 | Vadim Reutskiy | |
| v. 56 | Sep 09, 2020 05:32 | Vadim Reutskiy | Added details to the FR0903 |
| v. 55 | Sep 09, 2020 04:36 | Vadim Reutskiy | Added response and measure to the NFR0002 |
| v. 54 | Sep 08, 2020 11:30 | Vadim Reutskiy | |
| v. 53 | Sep 04, 2020 05:49 | Vadim Reutskiy | |
| v. 52 | Sep 04, 2020 05:48 | Vadim Reutskiy | |
| v. 51 | Sep 04, 2020 05:47 | Vadim Reutskiy | Add FR0113 - sending instructions with payload |
| v. 50 | Sep 04, 2020 05:34 | Vadim Reutskiy | |
| v. 49 | Sep 03, 2020 11:26 | Vadim Reutskiy | |
| v. 48 | Sep 03, 2020 10:46 | Vadim Reutskiy | |
| v. 47 | Sep 03, 2020 10:26 | Vadim Reutskiy | |
| v. 46 | Sep 03, 2020 10:00 | Vadim Reutskiy | |
| v. 45 | Sep 03, 2020 09:23 | Vadim Reutskiy | |
| v. 44 | Sep 03, 2020 07:03 | Vadim Reutskiy | Added section for high-level use cases and FR0900 and FR0901 about the fees configuration ar |
| v. 43 | Sep 03, 2020 05:32 | Vadim Reutskiy | |
| v. 42 | Sep 03, 2020 05:27 | Vadim Reutskiy | Added use case for selecting list of transactions FR0214 |

| | | | |
|-------|--------------------|------------------|--|
| v. 41 | Sep 03, 2020 05:19 | Vadim Reutskiy | Update list of participants and roles |
| v. 40 | Sep 03, 2020 05:16 | Vadim Reutskiy | |
| v. 39 | Sep 03, 2020 05:14 | Vadim Reutskiy | Added use cases for CRUD operations of data associated with the account: FR0212, FR0213, F |
| v. 38 | Sep 03, 2020 04:54 | Vadim Reutskiy | |
| v. 37 | Sep 02, 2020 08:03 | Vadim Reutskiy | |
| v. 36 | Sep 02, 2020 07:13 | Vadim Reutskiy | |
| v. 35 | Sep 02, 2020 05:53 | Vadim Reutskiy | Added FR0211 |
| v. 34 | Sep 02, 2020 05:31 | Vadim Reutskiy | |
| v. 33 | Sep 02, 2020 05:24 | Vadim Reutskiy | Added trigger-related use cases: FR0300-FR0302 |
| v. 32 | Sep 01, 2020 10:43 | Salakhiev Kamil | |
| v. 31 | Aug 30, 2020 10:41 | Vadim Reutskiy | |
| v. 30 | Aug 30, 2020 10:01 | Vadim Reutskiy | |
| v. 29 | Aug 30, 2020 08:49 | Vadim Reutskiy | |
| v. 28 | Aug 30, 2020 08:45 | Vadim Reutskiy | |
| v. 27 | Aug 30, 2020 08:43 | Vadim Reutskiy | |
| v. 26 | Aug 27, 2020 10:44 | Salakhiev Kamil | |
| v. 25 | Aug 27, 2020 10:25 | Bogdan Mingela | |
| v. 24 | Aug 27, 2020 06:56 | Vadim Reutskiy | |
| v. 23 | Aug 27, 2020 05:52 | Vadim Reutskiy | Added "source" field to all use cases |
| v. 22 | Aug 26, 2020 08:52 | Bogdan Mingela | |
| v. 21 | Aug 26, 2020 08:19 | Iurii Vinogradov | |
| v. 20 | Aug 26, 2020 08:14 | Iurii Vinogradov | |
| v. 19 | Aug 26, 2020 08:02 | Salakhiev Kamil | |

| | | | |
|-------|--------------------|---------------------|--|
| v. 18 | Aug 26, 2020 08:00 | Salakhie v Kamil | |
| v. 17 | Aug 25, 2020 09:57 | Vadim Reutskiy | Added FR0208, Subscribing on the query results |
| v. 16 | Aug 25, 2020 09:46 | Vadim Reutskiy | |
| v. 15 | Aug 25, 2020 09:45 | Vadim Reutskiy | |
| v. 14 | Aug 25, 2020 09:45 | Vadim Reutskiy | |
| v. 13 | Aug 25, 2020 09:41 | Vadim Reutskiy | |
| v. 12 | Aug 25, 2020 09:41 | Bogdan Mingela | blocks and pending transactions cases |
| v. 11 | Aug 25, 2020 07:01 | Vadim Reutskiy | |
| v. 10 | Aug 25, 2020 06:42 | Vadim Reutskiy | |
| v. 9 | Aug 25, 2020 05:53 | Vadim Reutskiy | |
| v. 8 | Aug 25, 2020 05:30 | Vadim Reutskiy | |
| v. 7 | Aug 25, 2020 05:06 | Vadim Reutskiy | |
| v. 6 | Aug 25, 2020 05:04 | Vadim Reutskiy | |
| v. 5 | Aug 24, 2020 11:26 | Vadim Reutskiy | |
| v. 4 | Aug 24, 2020 11:24 | Vadim Reutskiy | |
| v. 3 | Aug 24, 2020 11:22 | Vadim Reutskiy | |
| v. 2 | Aug 24, 2020 07:05 | Vadim Reutskiy | |
| v. 1 | Aug 24, 2020 06:00 | Vadim Reutskiy | |

Requirements

Actors

| Actor name | Role definition |
|-------------------|---|
| The Iroha peer | One of active peers of the Iroha network, accessible by network for the current user. |
| The administrator | The user of the Iroha network with the extended rights towards manipulating the peers and network configuration |
| The user | Generic user of the Iroha network with common list of permissions (required permissions are always mentioned in the use case Preconditions) |

Functional requirements

For describing functional requirements, we should follow the default use case template by example:

| | |
|------------------|---|
| Use case title | [FR0000] Example use-case; ID should be unique |
| Status | <div>DISCUSS</div> <div>REVIEWED</div> <div>DECIDED</div> <div>POSTPONE</div> |
| Source | Source (or list of sources) of the function: project name / stakeholder name / document title (e.g., whitepaper) / etc. |
| Preconditions | <ul style="list-style-type: none"> List of preconditions which must be satisfied before the use case can be applied Each precondition should be defined as the point in the list |
| Use case flow | <ol style="list-style-type: none"> The enumerated list of actions, which describes the interaction between actors step by step Each step should involve a single actor as an object of the action Each step can involve one or more actors as a subject of the action Each use case can involve other use cases as dependencies by include or extend relationships |
| Post-conditions | <ul style="list-style-type: none"> List of post-conditions, one or more results of successfully finished use case |
| Alternative flow | <ul style="list-style-type: none"> List of alternative flows that can happen with the use case. Each entry should have the link to the step of the main use case flow when the alternative can occur <ul style="list-style-type: none"> If needed, each entry can contain sub-entries Each entry should describe the consequences of the alternative flow |
| Exception flow | <ul style="list-style-type: none"> List of exceptions that can happen during the use case flow, which can prevent it to be successfully finished Each entry should have the link to the step of the main use case flow when the exception can occur <ul style="list-style-type: none"> If needed, each entry can contain sub-entries Each entry should describe the consequences of the exception |



Use case ID formula

Assuming, that each section would not have more than 100 use cases, the use case ID should be formed using that template:

FR<section_number> <use_case_number_in_section_starting_from_zero>

For example, for second use case in section 02 it should be:

FR02 01

00. Iroha network operations



In this section all use cases related to generic operations with Iroha network and peers will be described

| | |
|----------------|---|
| Use case title | [FR0001] Starting the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Generic blockchain functionality (Vadim Reutskiy) |
| | |

| | |
|------------------|---|
| Preconditions | <ul style="list-style-type: none"> The administrator has configured environment for running Iroha executable The administrator has prepared the information for the network configuration (peer addresses and ports, consensus parameters, etc.) The administrator generated key pairs for all accounts, which should be created at the start of the network, including root administrator account if it is required by business requirements The administrator has prepared the genesis block configuration according to the network configuration and business requirements |
| Use case flow | <ol style="list-style-type: none"> The administrator launches the command to run Iroha peer and providing the path to the genesis block description The Iroha peer read and validates the genesis block configuration The Iroha peer starts the Iroha network according to the configuration in the genesis block The Iroha peer provides "successfully started" report to the administrator |
| Post-conditions | <ul style="list-style-type: none"> The Iroha network is running The administrator has access to the root account of the network |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> At step 2, in case of the invalid genesis block, the Iroha peer shows corresponding error messages to the administrator and halts the network creation process |

| | |
|-----------------------|---|
| Use case title | [FR0002] Adding peer to the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Generic functionality (Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The Iroha network is running The administrator has access to the account with permissions of adding peers to the network |
| Use case flow | <ol style="list-style-type: none"> The administrator configures and starts the new peer The administrator prepares the transaction with instruction to add the new peer The administrator sends the prepared transaction to the existing Iroha peer (<<include>> FR0100) The existing Iroha peer initiates connection with the new Iroha peer The new Iroha peer synchronizes the block-storage with existing Iroha peers The new Iroha peer start functioning as fully functional peer in the current Iroha network |
| Post-conditions | <ul style="list-style-type: none"> The new peer is added into the network |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 5, if the synchronization cannot be finished for some reason, the new Iroha peer cannot join the network; use case stops |

| | |
|-----------------------|--|
| Use case title | [FR0003] Removing peer from the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Basic blockchain functionality (Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The administrator has access to the account with permission of removing peers form the network |
| Use case flow | <ol style="list-style-type: none"> The administrator prepares the instruction to remove the peer from the Iroha network |

| | |
|------------------|---|
| | <ol style="list-style-type: none"> 2. The administrator sends the transaction with the prepared instruction to the Iroha peer 3. The Iroha peer confirms the execution of the instruction |
| Post-conditions | <ul style="list-style-type: none"> • The target peer is removed from the current Iroha network and all processes in consensus continues without it |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0004] Configuring initial state of the Iroha network |
| Status | DISCUSS |
| Source | |
| Preconditions | <ul style="list-style-type: none"> • The Iroha network is just started |
| Use case flow | TBD - questionable functionality |
| Post-conditions | <ul style="list-style-type: none"> • The Iroha network is configured according to the requirements |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0005] Changing configuration of working Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> • The administrator has access to account with permissions to change the network configuration by sending the corresponding instruction |
| Use case flow | <ol style="list-style-type: none"> 1. The administrator prepares the data for the instruction for changing the overall Iroha network configuration. Configuration may include <ol style="list-style-type: none"> a. Default time to life (TTL) for pending multi-signature transaction b. Maximum transactions in the single block c. Maximum time of block filling by transactions (in milliseconds) d. TBD 2. The administrator sends the transaction with prepared instruction to the Iroha peer (<<include>> FR0100) 3. The Iroha peer return the results of instruction execution to the administrator. |
| Post-conditions | <ul style="list-style-type: none"> • The network configuration is changed correspondingly |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0006] Changing configuration of the particular peer |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Generic peer functionality (Vadim Reutskiy) |
| | |

| | |
|------------------|---|
| Preconditions | <ul style="list-style-type: none"> The administrator has access to the maintenance endpoint of the target Iroha peer |
| Use case flow | <ol style="list-style-type: none"> The administrator connects to the maintenance endpoint of the target Iroha peer The administrator performs requests to change required parameters of the target Iroha peer The Iroha peer returns result of change operation to the administrator |
| Post-conditions | <ul style="list-style-type: none"> The configuration of the target Iroha peer was changed |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the value of changed configuration parameter is out of the limit, the Iroha peer will return "wrong parameter value" error status to the administrator; use case flow stops. |

01. Making changes in Iroha network data by Iroha special instructions



In this section all use cases related to changing data in the Iroha network will be described

| | |
|------------------|---|
| Use case title | [FR0100] Sending the transaction to the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 whitepaper (Makoto Takemiya) |
| Preconditions | <ul style="list-style-type: none"> The user has an account in the Iroha network The user has prepared the transaction for sending; transactions can consist of several instructions The user has permissions to execute all instruction included in the transaction The Iroha network is working in normal condition |
| Use case flow | <ol style="list-style-type: none"> The user sends the transaction to the Iroha peer through the API The Iroha peer receives the transaction and validates its contents The Iroha peer checks that the user's account have enough permissions to run all instructions included in the transaction The Iroha peer executes the transaction by applying it to the current block The Iroha peer returns the result of the operation to the user |
| Post-conditions | <ul style="list-style-type: none"> The transaction is successfully executed and applied to the network state |
| Alternative flow | <ul style="list-style-type: none"> On step 4, if the current user's account is multi-signature, the result of the transaction will be applied to the block-store only when the number of signatures will reach the current <i>quorum</i> value. In other cases, the Iroha peer will return "pending" status as a response on step 5. |
| Exception flow | <ul style="list-style-type: none"> On step 2, one of the instructions has invalid structure, the Iroha peer returns an error message with description to the user and stops the flow On step 3, if the user's account does not have required permissions to execute one of the instructions, the Iroha peer returns information about that error and stops the flow On step 3, in case of a <i>multi-signature transaction</i>, if the instruction signed using the key pair, which was already used for signing that instruction, the Iroha peer will return "already signed" error status and stop the flow |
| Use case title | [FR0101] Creation of the account in the Iroha network |

| | |
|------------------|--|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 whitepaper (Makoto Takemiya) |
| Preconditions | <ul style="list-style-type: none"> The user has permission to create accounts |
| Use case flow | <ol style="list-style-type: none"> The user prepares information about the account that should be created, which includes name, domain, the public key and permissions list The user prepares instruction to send to the Iroha network according to the prepared information The user sends the transaction with the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns the result of the instruction execution to the user |
| Post-conditions | <ul style="list-style-type: none"> The new account in the Iroha network is created |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the account with the provided name or public key already exists in the network, the Iroha peer will return "already exists" error status; user case flow will stop. |

| | |
|-----------------------|--|
| Use case title | [FR0102] Configuring permissions for the account in the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 whitepaper (Makoto Takemiya) Sora 2 (by Zilya Yagafarova and Pavel Golovkin) Bakong (by Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> The user has access to the Iroha account with permissions to change permissions of the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares information about set of permissions, which should be enabled or disabled for the target account The user prepares instruction to send to the Iroha network according to the prepared information The user sends the transaction with the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns the result of the instruction execution to the user |
| Post-conditions | <ul style="list-style-type: none"> The permissions of the target account was changed correspondingly |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 4, if the user disables the permission which is not enables on the target account, the Iroha peer return "Permission is not enabled" error status; use case flow stops On step 4, if the user enables the permission which is already enabled on the target account, the Iroha peer return "Permission already enabled" error status; use case flow stops |

| | |
|-----------------------|--|
| Use case title | [FR0102] Granting permissions for the account in the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Bakong (by Zilya Yagafarova) |
| Preconditions | |

| | |
|------------------|---|
| | <ul style="list-style-type: none"> • The user has access to his/her account in the Iroha network • The user's account has permissions to provide grantable permissions |
| Use case flow | <ol style="list-style-type: none"> 1. The user prepares the instruction to grant one or more permissions to another account 2. The user sends the transaction with the prepared instruction to the Iroha peer (<<includes>> FR0100) 3. The Iroha peer returns the result of the instruction to the user |
| Post-conditions | <ul style="list-style-type: none"> • The permission was granted to another account |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> • On step 3, if the user does not have permission to grant permissions to other accounts, the Iroha peer will return "Not permitted" error status; use case flow stops |

| | |
|-----------------------|---|
| Use case title | [FR0104] Sending complex instruction using ISI DSL |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Iroha 2 whitepaper (Makoto Takemiya) |
| Preconditions | <ul style="list-style-type: none"> • The user has permissions to send complex instructions with DSL sequence inside • The user has permissions to execute each instruction inside the list of DSL sequence |
| Use case flow | <ol style="list-style-type: none"> 1. The user builds the DSL sequence for the complex instruction 2. The user prepares the complex instruction to send to the Iroha peer 3. The user sends the complex instruction to the Iroha peer (<<include>> FR0100) 4. The Iroha peer returns the results of complex instruction execution to the user |
| Post-conditions | <ul style="list-style-type: none"> • The complex instruction was executed and applied to the Iroha network |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> • On step 4, if there is not enough permissions to execute whole DSL sequence, the Iroha peer will return "not enough permissions" error status; use case flow stops. |

| | |
|-----------------------|---|
| Use case title | [FR0105] Sending instruction and subscribing to the status of finalization |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • The Sora project (by Ruslan Rezin) |
| Preconditions | <ul style="list-style-type: none"> • The user has permissions to execute requested instruction |
| Use case flow | <ol style="list-style-type: none"> 1. The user prepares the instruction parameters according to his/her needs 2. The user sends the transaction with the instruction into the Iroha peer (<<includes>> FR0100) 3. The user uses the same connection for obtaining the status of the instruction finalization 4. The Iroha peer sends updates about the instruction finalization status to the user 5. The user eventually receives the "successfully finalized" status about the instruction |
| Post- | |

| | |
|------------------|---|
| conditions | <ul style="list-style-type: none"> • The instruction is accepted and finalized in the block-storage • The user did not spend additional resources for redundant network connections and requests |
| Alternative flow | <ul style="list-style-type: none"> • At step 5, if there is some problem with instruction or Iroha network the "successfully finalized" may never be obtained; in that case, the behavior of client depends on the project business rules and particular requirements. |
| Exception flow | <ul style="list-style-type: none"> • At step 4 if the instruction cannot pass the validation, the user would not get the "successfully finalized" status; instead, he/she will get the "invalid instruction" status. • At step 4 if the network is in "bad" condition, the user would get the "successfully finalized" status eventually, when the network will return in a "good" state. |

| | |
|------------------|--|
| Use case title | [FR0106] Creation of the multi-signature account in the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Iroha 2 whitepaper (Makoto Takemiya) • Sora 2 project (by Zilya Yagafarova and Pavel Golovkin) • Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> • The user have permission to create new MST accounts |
| Use case flow | <ol style="list-style-type: none"> 1. The user prepares information about new account, including list of signatories and quorum of signatures 2. The user creates new account in the Iroha network (<<include>> FR0101) 3. The user receives result of new account creation from the Iroha peer |
| Post-conditions | <ul style="list-style-type: none"> • The new MST account with defined list of signatories and quorum is created |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> • On step 3, if amount of signatories in the list is lower, than quorum, the Iroha peer returns "Not enough signatories" error; use case flow stops |

| | |
|------------------|---|
| Use case title | [FR0107] Changing quorum for the multi-signature account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> • The user have access to the account with permissions to control the target MST account |
| Use case flow | <ol style="list-style-type: none"> 1. The user prepares the instruction with needed parameters for changing the quorum value 2. The user sends the transaction to the Iroha peer (<<include>> FR0100) 3. The Iroha peer returns the results of instruction execution |
| Post-conditions | <ul style="list-style-type: none"> • The quorum for the target MST account was changed |
| Alternative flow | N/A |
| | |

| | |
|----------------|---|
| Exception flow | <ul style="list-style-type: none"> If the new quorum is higher than amount of signatories, the Iroha peer will return "Not enough signatories" error message; use case flow stops. |
|----------------|---|

| | |
|------------------|---|
| Use case title | [FR0108] Changing list of signatories for the multi-signature account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to change the list of signatories to the required account The user has key pair for adding/deleting a signatory to/from the selected account |
| Use case flow | <ol style="list-style-type: none"> The user prepares the instruction of adding/deleting signatories for sending The user sends the transaction with the instruction to the Iroha peer (<<include>> FR0100) The user receives the response from the Iroha peer |
| Post-conditions | <ul style="list-style-type: none"> List of signatories for the selected account was changed correspondingly |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 2, if the signatory with the current key pair exists in the list of signatories in the target account, the Iroha peer will return "already added" error status; use case flow stops. |

| | |
|------------------|--|
| Use case title | [FR0109] Signing multi-signature transaction |
| Status | DISCUSS |
| Source | <ol style="list-style-type: none"> Iroha 2 whitepaper Internal project by Iurii Vinogradov |
| Preconditions | <ul style="list-style-type: none"> There is a multi-signature account in the Iroha network The user has access to the one or many key pairs for the multi-signature account The current amount of signatures to the multi-signature transaction is lower than a current quorum value |
| Use case flow | <ol style="list-style-type: none"> The user obtains the current list of pending transactions (<<include>> FR0203) The user finds the required transaction that can be signed by his/her signatures The user signs required transaction using the available key pair by calling the corresponding method from CLI or client SDK The user sends signed transaction to the Iroha peer (<<include>> FR0100) |
| Post-conditions | <ul style="list-style-type: none"> Amount of signatures in the signed instruction increased by 1 signature Signed instruction disappears from the list of pending instructions for the current user's account |
| Alternative flow | <ul style="list-style-type: none"> On step 2 the user can select more than one transactions, all of them can be sent on step 4 as a separate requests The user can skip steps 1 and 2 if he/she already has information about the pending transaction, which was provided by any external integration. Hence, the user need to have possibility to start the use case directly from the step 3 by sending signed transaction without prior queries. (by Iurii Vinogradov) On step 3 the user can sign the transaction by more than one signature, and all of them can be sent as a single request per single transaction on step 4. In bridge implementation scenario is: 1. Selected bridge get all bridge instances signatures from the block and calls an Iroha2 ISI with the transfer information, attaching all bridge nodes signatures. It can be implemented as in Iroha1 by sending multiple ISI calls in one transaction or batch, or it can be implemented as One ISI call with multiple signatures. Iurii Vinogradov |

| | |
|--------------------|--|
| Excepti on flow | <ul style="list-style-type: none"> On step 1, in case of the empty list of pending transactions, the Iroha peer will return corresponding status and the use case flow stops On step 3, if the user uses key pair, which was already used for signing the transaction, the CLI/SDK will return corresponding error status; use case flow stops if there are no more available key pairs. TBD (define, will it be available to do on the client-side?) |
|--------------------|--|

| | |
|------------------|---|
| Use case title | [FR0110] Changing the conditions for the multi-signature account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project Bakong project Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to change the conditions for the target multi-signature account |
| Use case flow | <ol style="list-style-type: none"> The user requests the list of currently configured conditions for the target multi-signature account The user prepared the instruction with the list of conditions to be added/removed The user sends the transaction with the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer sends the response to the user |
| Post-conditions | <ul style="list-style-type: none"> The list of conditions for the target multi-signature account was changed correspondingly |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|------------------|---|
| Use case title | [FR0111] Assigning weights to the signatories of the multi-signature account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Bogdan Mingela Sora 2 project (future versions) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to change the conditions (or some other, to discuss) for the target multi-signature account |
| Use case flow | <ol style="list-style-type: none"> The user prepared the instruction with the map of public keys and weight correspondence (e.g. key1 50, i.e. key2 25, key3 25) and the target quorum (e.g. 50 so that it is needed either to sign upcoming transactions of the user with either key1 or both key2 and key3) The user sends the transaction with the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer sends the response to the user |
| Post-conditions | <ul style="list-style-type: none"> The list of signatories weights and target quorum for the target multi-signature account was changed correspondingly |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|----------------|---|
| Use case title | [FR0112] Associating and changing arbitrary data payload with the account |
|----------------|---|

| | |
|------------------|--|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project (Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> The user has access to account with permissions to change data payload related to the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares the key and value of the data payload for forming the corresponding instruction The user sends the transaction with prepared instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns results of instruction to the user |
| Post-conditions | <ul style="list-style-type: none"> The data payload associated with the target account was changed correspondingly |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can define the previous value of the selected key, so the Iroha peer will compare that value with the current value saved in the block storage on step 2. If values will be different, the Iroha peer will return error status on step 3. On step 2, if there was no such key associated with the target account, the key will be created On step 2, if some value is already associated with the provided key, the value will be overwritten |
| Exception flow | <ul style="list-style-type: none"> On step 2, if the size of payload data exceeds limit, the Iroha peer will return the "Too large data" error message; use case flow stops. |

| | |
|------------------|---|
| Use case title | [FR0113] Sending instruction with the payload |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Makoto Takemiya Sora 2 project Internal project by Iurii Vinogradov |
| Preconditions | <ul style="list-style-type: none"> The user has access to account with permissions to perform required instruction |
| Use case flow | <ol style="list-style-type: none"> The user prepares the instruction for the required operation and attached the payload to the instruction The user sends the transaction with prepared instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns results of instruction execution to the user |
| Post-conditions | <ul style="list-style-type: none"> The instruction with attached payload successfully executed and payload saved in the block storage |
| Alternative flow | <ul style="list-style-type: none"> On step 2, there should be an option to verify the payload before applying the instruction ("compare and set" operation); in case if the comparison fails, the instruction should not be applied. |
| Exception flow | <ul style="list-style-type: none"> On step 2, if the size of payload data exceeds limit, the Iroha peer will return the "Too large data" error message; use case flow stops. |

| | |
|----------------|--|
| Use case title | [FR0114] Sending non-fungible assets |
| Status | DISCUSS - may be postponed |
| Source | <ul style="list-style-type: none"> Bakong project |

| | |
|------------------|---|
| Preconditions | <ul style="list-style-type: none"> The sender has access to the account in the Iroha network There is at least one non-fungible asset type in the Iroha network There are at least one token of non-fungible token type on the sender account |
| Use case flow | <ol style="list-style-type: none"> The sender requests list of currently available non-fungible assets on the personal account by sending query to the Iroha network (<<include>> FR02016) The sender prepares the information about non-fungible assets which should be sent <ol style="list-style-type: none"> Information must include IDs of assets which need to be sent The sender sends the transaction with "send" instruction to the Iroha network (<<include>> FR0100) The Iroha network returns results of the transaction to the sender |
| Post-conditions | <ul style="list-style-type: none"> The sender sent non-fungible token(s) to the recipient |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can configure filtering, so the Iroha peer will return the list of all assets which are correspond to the filters. |
| Exception flow | N/A |

02. Acquiring data from the Iroha network by queries



In this section all use cases related to retrieving data from the Iroha network will be described

| | |
|------------------|--|
| Use case title | [FR0200] Acquiring data from the Iroha network by query |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 whitepaper Generic functionality of Iroha |
| Preconditions | <ul style="list-style-type: none"> User has access to the account with permissions, which allows to get results of target query |
| Use case flow | <ol style="list-style-type: none"> The user sends the query to the Iroha peer through the API The Iroha peer receives the query and validates its contents The Iroha peer checks that the user's account have enough permissions to get all data requested in query The Iroha peer collects information requested in query from the current WSV The Iroha peer creates Merkle tree for the obtained data TBD The Iroha peer returns the result of the operation to the user |
| Post-conditions | <ul style="list-style-type: none"> User receives the response with the requested data |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 2, if the query was formed not correctly (for example, because of problems with the client library) the Iroha peer returns error with result of validation, use case execution stops. On step 3, if the user's account does not have enough permissions, the Iroha peer returns error with corresponding status, use case execution stops. |
| Use case title | [FR0201] Acquiring the information about the selected account |

| | |
|------------------|--|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 white paper |
| Preconditions | <ul style="list-style-type: none"> The user has access to the account with permissions of getting information about the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for requesting information about the target account The user sends the query to the Iroha peer (<<include>> FR0200) The user receives the request with the list of parameters of the target account <ol style="list-style-type: none"> It should include: account name, domain, permissions, quorum, list of signatories, associated key-value dictionary with payload |
| Post-conditions | <ul style="list-style-type: none"> The user received the information about the selected account |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0202] Acquiring of the current permissions for the selected account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 white paper |
| Preconditions | <ul style="list-style-type: none"> The user has access to the account with permissions of checking list of permissions of the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for requesting a list of permissions for the target account The user sends the query to the Iroha peer (<<include>> FR0200) The user receives the request with the list of permissions for the target account |
| Post-conditions | <ul style="list-style-type: none"> The user received the list of permissions for the target account |
| Alternative flow | N/A |
| Exception flow | N/A |

| | | |
|-----------------------|---|--|
| Use case title | [FR0203] Acquiring a list of pending multi-signature transactions | |
| Status | DISCUSS | |
| Source | <ul style="list-style-type: none"> Sora 2 project | |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to request the list of pending transactions | |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for requesting the list of the pending transaction <ol style="list-style-type: none"> The user can request the list of all pending transaction in the system (if there are according permissions enabled for the account). The user can request the list of pending transactions created by the current account. The user can request the list of pending transactions from other accounts which requires the user's key pair to be signed. The user sends the query to the Iroha peer (<<include>> FR0200) The user receives the list of currently pending multi-signature transaction TBD (should the Iroha expose all pending transactions, or we need to add the account ID / public key to filter the request?) | |

| | |
|------------------|--|
| Post-conditions | <ul style="list-style-type: none"> The user has the list of currently pending instructions according to provided parameters |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can request all pending transactions On step 1, the user can request pending transactions, which waits his/her signature only (by account ID or by providing public key from the target key pair) |
| Exception flow | <ul style="list-style-type: none"> On step 3, if there are no currently pending transactions, the Iroha peer will return "empty response" status |

| | |
|------------------|--|
| Use case title | [FR0204] Acquiring a list of current conditions for a multi-signature account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Bakong project Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to request a list of conditions to the target multi-signature account |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for requesting a list of currently configured conditions of signing for the target multi-signature account The user sends the query to the Iroha peer (<<include>> FR0200) The user receives the list of currently configured conditions of signing for the target multi-signature account |
| Post-conditions | <ul style="list-style-type: none"> The user received the list of conditions for the target account |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if there are no currently configured conditions, the Iroha peer will return "empty response" status On step 3, if the account is not multi-signature, the Iroha peer will return "not applicable" status |

| | |
|------------------|---|
| Use case title | [FR0205] Acquiring a block by its number |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Bogdan Mingela Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to request the block by its number |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for requesting the block with defined number The user sends the query to the Iroha peer (<<include>> FR0200) The user receives the block description with all its' transactions, etc. |
| Post-conditions | <ul style="list-style-type: none"> The user receives the requested block from the block-store |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if there are no block specified, the Iroha peer will return "error/no such block" response |

| | |
|----------------|--|
| Use case title | [FR0206] Acquiring blocks subscription |
|----------------|--|

| | |
|------------------|---|
| Status | DISCUSS (can be extended with a start block number index) |
| Source | Bogdan Mingela |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to request blocks (subscription, can be separate permission or from <<include>> FR0205) |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for subscription of blocks The user establishes the subscription to the query results with the Iroha peer (<<include>> FR0208) The user analyses the results of continuously received payload |
| Post-conditions | <ul style="list-style-type: none"> The user has got stable channel for continuous block obtaining, which can be used for getting all new blocks on the current peer until the connection will be interrupted |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0207] Acquiring pending transactions subscription |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora project (by Bogdan Mingela) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to request the list of pending transactions (subscription, can be separate permission or from <<include>> FR0203) |
| Use case flow | <ol style="list-style-type: none"> The user prepares query for subscription for pending transactions The user established the subscription with the Iroha peer (<<include>> FR0208) The user gets the connection to receive pending transactions within initialized |
| Post-conditions | <ul style="list-style-type: none"> The user has got stable pending transactions perception channel and newly arriving pending transactions |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0208] Subscribing on the query results |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora project (by Ruslan Rezin) Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to get results of the target query |
| Use case flow | <ol style="list-style-type: none"> The user prepares request data for the target query The user establishes permanent connection to the Iroha peer and sends the query The Iroha peer confirms the connection and start sending updates on the query results until the user closes the connection The user receives the results and performs continuous analysis of the payload |

| | |
|------------------|--|
| Post-conditions | <ul style="list-style-type: none"> The user has continuous results of query execution |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the Iroha peer experiences issues with the network connection to other peers, the flow of results may be unplugged TBD On step 3, if the user experiences issues with the network connection to the Iroha peer, the flow of results will be interrupted On step 4, if permissions of current account, which the user uses for establishing subscription, changes and not longer allows to get the data requested by query, the Iroha peer interrupts the subscription; use case flow stops. |

| | |
|------------------|--|
| Use case title | [FR0209] Validate result of the query |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Salakhiev Kamil Sora project (by Ruslan Rezin , Anton Khvorov, Zilya Yagafarova , Pavel Golovkin) Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to get results of the target query |
| Use case flow | <ol style="list-style-type: none"> The user prepares request data for the target query The user sends the query to the Iroha peer (<<include>> FR0200) The Iroha peer returns response containing result of the query as well as the Merkle proof of its validity and relation to the current block-store. The user receives the response and validates the correctness of the data by comparing hashes in Merkle proof and hashes of obtained data. |
| Post-conditions | <ul style="list-style-type: none"> The user has result of the query request with Merkle proof of its validity |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 4, if the response was corrupted or changed, the user can find it by comparing results of data hashing with the Merkle proof. In that case, validation can be considered as failed. |

| | |
|-----------------|---|
| Use case title | [FR0210] Query old Iroha state (e.g., query balance month ago) |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha community (by Salakhiev Kamil) Bakong (by Zilya Yagafarova) Sora 2 (by Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> The user has access to account with permissions to get requested data at the current moment TBD The user has access to account which had permissions to get requested data at the moment in the past, requested by the user |
| Use case flow | <<extends>> FR0200, substitutes step 4: <ol style="list-style-type: none"> The Iroha peer collects data from the block storage at the moment, requested in the query |
| Post-conditions | <ul style="list-style-type: none"> The user received the requested information about the information in block store on the provided date |

| | |
|------------------|--|
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 1, if there is no data for the requested query on the required date, the Iroha peer will return error response with corresponding state |

| | |
|-----------------------|--|
| Use case title | [FR0211] Querying list of accounts with the predefined filter |
| Status | DISCUSS - probably optional for MVP, if there will be another way for gathering data for the award calculation for liquidity providers in DEX solution |
| Source | <ul style="list-style-type: none"> Internal project by Iurii Vinogradov |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to perform the query The user has list of conditions which need to be used for filtering accounts |
| Use case flow | <ol style="list-style-type: none"> The user prepares the query and adds the filtering setup to the query body The user sends query to the Iroha peer (<<include>> FR0200) The Iroha peer returns result of the query execution to the user |
| Post-conditions | <ul style="list-style-type: none"> The user has the list of accounts which satisfies entered filter |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can include the query to the DSL sequence as a part of the complex instruction |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the user does not have enough permissions, the Iroha peer returns "not enough permission" error status; the use case flow stops. |

| | |
|-----------------------|--|
| Use case title | [FR0212] Retrieving the list of keys of data payload, associated with the target account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova by Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The user has access to account with permissions to retrieve data payload related to the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares the target account identifier for forming the corresponding query The user sends the transaction with prepared query to the Iroha peer (<<include>> FR0200) The Iroha peer returns results of query to the user, which contains the list of keys, associated with the target account |
| Post-conditions | <ul style="list-style-type: none"> The user acquired list of keys of data payload associated with the target account |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 2, if there is no keys associated with the target account, the Iroha peer will return "No data" status message; use case flow stops. |

| | |
|-----------------------|--|
| Use case title | [FR0213] Retrieving the value of data payload by the key, associated with the target account |
| | |

| | |
|------------------|--|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova by Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The user has access to account with permissions to retrieve data payload related to the target account |
| Use case flow | <ol style="list-style-type: none"> The user prepares the target account identifier and required key for forming the corresponding query The user sends the transaction with prepared query to the Iroha peer (<<include>> FR0200) The Iroha peer returns results of query to the user, which contains value of data payload, associated with target key on target account |
| Post-conditions | <ul style="list-style-type: none"> The user acquired value of data payload, associated with target key on target account |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 2, if there is no such keys associated with the target account, the Iroha peer will return "Wrong key" error message; use case flow stops. |

| | |
|-----------------------|---|
| Use case title | [FR0214] Querying list of transactions with predefined filter |
| Status | DISCUSS - probably optional for MVP, if there will be another way for gathering data for the getting list of all transactions |
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova and Pavel Golovkin by Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to perform the target query The user has list of conditions which need to be used for filtering transactions (optionally) |
| Use case flow | <ol style="list-style-type: none"> The user prepares the query for getting list of transactions and adds the filtering data to the query body <ol style="list-style-type: none"> Filtering may filter out incoming or outgoing transactions Filtering may filter out transactions by receiver and/or sender account ID Filtering may filter out transactions by time frame The user sends query to the Iroha peer (<<include>> FR0200) The Iroha peer returns result of the query execution to the user |
| Post-conditions | <ul style="list-style-type: none"> The user has the list of transactions which satisfies the predefined filter with pagination |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the filtering data can be optional; in that case the Iroha peer will return list of all transactions without any filtering On step 1, the user can also include paging-related request into the query, in that case the Iroha peer will return the data correspondingly |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the user does not have enough permissions, the Iroha peer returns "not enough permission" error status; the use case flow stops. On step 3, if the user provided setup of filtering, which results in empty response, the Iroha peer will return "Empty response" status message; use case stops, On step 3, if the user provided incorrect pagination-related request part, the Iroha peer will return "Incorrect pagination request" error message; use case stops. |

| | |
|-----------------------|--|
| Use case title | [FR0215] Subscription on incoming transactions |
| Status | DISCUSS |

| | |
|------------------|--|
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova and Pavel Golovkin) Bakong (from Zilya Yagafarova) Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to perform the target query of getting list of incoming transactions The user has list of conditions which need to be used for filtering transactions |
| Use case flow | <ol style="list-style-type: none"> The user prepares the query for getting list of incoming transactions and adds the filtering data to get only incoming transactions The user sends query to the Iroha peer and subscribes on the results (<<include>> FR0208) The Iroha peer continuously returns result of the query execution to the user The user processes the incoming results of the query Eventually, the user interrupts the subscription with the Iroha peer, which stops the flow of data about incoming transactions |
| Post-conditions | <ul style="list-style-type: none"> The user has all incoming transactions during the period when the subscription was active |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can configure list of accounts, which should be used as filter for the query |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0216] Requesting list of non-fungible assets in account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Bakong (from Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to perform the target query of getting list of incoming transactions |
| Use case flow | <ol style="list-style-type: none"> The user prepares the query for getting list of non-fungible assets The user sends query to the Iroha peer (<<include>> FR0200) The Iroha peer responses with result of the query execution to the user, which contains list of all non-fungible assets with IDs and parameters |
| Post-conditions | <ul style="list-style-type: none"> The user has a list with IDs and parameters of all non-fungible assets attached to the account |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the user can configure filtering for the request, so the resulting output on step 3 will contain only those assets, which are corresponds to the invariant |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0217] Requesting data from the block storage by using special request language |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> The user has permissions to perform query using special language The user has permissions to all data inside block storage, which he/she wants to request by the query |
| Use case flow | <ol style="list-style-type: none"> The user prepares the complex query by building request on the provided query language |

| | |
|------------------|--|
| | <ol style="list-style-type: none"> 2. The user sends prepared query to the Iroha peer (<<include>> FR0200) 3. The Iroha peer responds with result of the query execution to the user |
| Post-conditions | <ul style="list-style-type: none"> • The user requests all data according to the query language, which is available to his account according to permissions |
| Alternative flow | N/A |
| Exception flow | N/A |

03. Setting up and executing triggers



In this section all use cases related to triggers in the Iroha network will be described

| | |
|------------------|--|
| Use case title | [FR0300] Setting up a trigger in the Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Internal project by Iurii Vinogradov; manual execution will be enough for MVP |
| Preconditions | <ul style="list-style-type: none"> • The user has permissions to setting up triggers in the Iroha network • The user has permissions to execute all instructions and queries which are included into the trigger body |
| Use case flow | <ol style="list-style-type: none"> 1. The user prepares the list of instructions in form of DSL sequence 2. The user prepares the instruction for sending trigger body to the Iroha peer <ol style="list-style-type: none"> a. The user can configure trigger to fire in manual mode only, by sending specific instruction to the Iroha peer; b. The user can configure trigger to fire when the temporal condition becomes true; condition can be based on the block number or block timestamp; c. The user can configure trigger to fire when the data condition becomes true; condition can be based on the data available by permissions to the author user; TBD (potential problem with the performance) d. The user can configure the trigger to use the state of already existing trigger (if there are enough permissions on the user's account) 3. The user sends the instruction to the Iroha peer (<<include>> FR0100) 4. The Iroha peer returns result of trigger setup to the user, which includes ID of the trigger which can be used to manually fire the trigger in future 5. The Iroha peer checks the conditions of the triggers periodically; if the condition becomes true, the Iroha peer checks permissions of the author user and executes the trigger body |
| Post-conditions | <ul style="list-style-type: none"> • The trigger was configured and ready to fire • The user has ID of the created trigger to use it in future manipulations with it |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> • On step 2, the user can also explicitly state permissions of the trigger's account, in the aim to control the possibilities of the code which will be executed by firing the trigger. • On step 4, if the user does not have permissions to set up triggers and/or if there is not enough permissions to each command in the trigger body, the Iroha peer return "not enough permissions" error response; use case flow stops. • On step 5, if permissions of the author user would not allow for all instructions to execute (for example, if permissions of the author user was changed after creating the trigger), the whole trigger execution will be cancelled. TBD |

| | |
|----------------|--------------------------------------|
| Use case title | [FR0301] Manually firing the trigger |
| Status | DISCUSS |
| | |

| | |
|------------------|---|
| Source | <ul style="list-style-type: none"> Internal project by Iurii Vinogradov |
| Preconditions | <ul style="list-style-type: none"> The trigger was created and configured as "manual" by the FR0300 The user has permissions to fire the trigger The user has ID of the target trigger |
| Use case flow | <ol style="list-style-type: none"> The user prepares the instruction to fire the target trigger The user sends the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns result of the target trigger execution |
| Post-conditions | <ul style="list-style-type: none"> All instructions from the trigger body was executed and applied to the block storage |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if user does not have permissions to run all instructions in the Trigger, the Iroha peer returns "not enough permissions" status; use case flow stops. |

| | |
|-----------------------|---|
| Use case title | [FR0302] Removing the trigger |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Logically extrapolated from FR0300 by Vadim Reutskiy |
| Preconditions | <ul style="list-style-type: none"> The trigger was created by the FR0300 The user has permissions to remove the trigger The user has ID of the target trigger |
| Use case flow | <ol style="list-style-type: none"> The user prepares the instruction to remove the trigger The user sends the instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns results of the instruction to the user |
| Post-conditions | <ul style="list-style-type: none"> The target trigger was removed from the list of active triggers |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the user does not have enough permissions, the Iroha peer will return "not enough permissions" error status; use case flow stops |

09. High-level use cases



In this section all high-level use cases will be described

| | |
|-----------------------|---|
| Use case title | [FR0900] Configuration of fees for transfers inside the current Iroha network |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova and Pavel Golovkin by Vadim Reutskiy) |

| | |
|------------------|--|
| Preconditions | <ul style="list-style-type: none"> The network administrator have access to the account with permissions to configure fees in the current Iroha network |
| Use case flow | <ol style="list-style-type: none"> The network administrator prepares the business-level description of the fee configuration The network administrator sets up the fee configuration using the format (or DSL) used in Iroha The network administrator prepares the instruction to change fee in the Iroha network The network administrator sends the prepared instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns results of the instruction to the user |
| Post-conditions | <ul style="list-style-type: none"> The fee configuration was changes according to the provided data |
| Alternative flow | N/A |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the user does not have enough permissions, the Iroha peer will return "not enough permissions" error status; use case flow stops |

| | |
|-----------------------|---|
| Use case title | [FR0901] Sending transaction with fees |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project (from Zilya Yagafarova and Pavel Golovkin by Vadim Reutskiy) |
| Preconditions | <ul style="list-style-type: none"> The user has access to the account with permissions to transfer assets to other account |
| Use case flow | <ol style="list-style-type: none"> The user prepares the information about needed transfer of assets The user sends the query to the Iroha peer in the aim to get the needed structure of transaction, including all required fees (<<include>> FR0200) <ol style="list-style-type: none"> The response also contains the time frame until the provided information will be relevant. The user creates the transaction with all required instructions included within the provided time frame The user sends the prepared instruction to the Iroha peer (<<include>> FR0100) The Iroha peer returns results of the instruction to the user |
| Post-conditions | <ul style="list-style-type: none"> The transfer instruction was successfully applied, together with all needed instructions for fees deduction. |
| Alternative flow | <ul style="list-style-type: none"> On step 3, if the time frame has passed, the user needs to request the information about fees again in the aim to create correct transaction |
| Exception flow | <ul style="list-style-type: none"> On step 3, if user sends the transaction within the time frame, there is a chance that fees will be changed and the transaction will be rejected; use case flow stops. |

| | |
|-----------------------|---|
| Use case title | [FR0902] Delegation of account control with time limit |
| Status | DISCUSS - could be after MVP |
| Source | <ul style="list-style-type: none"> Bakong (from Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> The user has need to delegate full or partial control over the account to another account (optionally) The user wants to limit the delegation period by time |

| | |
|------------------|---|
| Use case flow | <ol style="list-style-type: none"> 1. The target user (which will get delegated rights on the account control) provides the account ID to the initial user 2. The initial user prepares the data for instruction for delegating control over the account 3. The initial user sends the transaction with instruction to the Iroha peer (<<include>> FR0100) 4. The Iroha peer executes the instruction and applies it to the block-store 5. The Iroha peer returns the response with the result of the instruction execution |
| Post-conditions | <ul style="list-style-type: none"> • The control over the initial user's account was delegated to the target account |
| Alternative flow | <ul style="list-style-type: none"> • On step 1, the initial user also can configure the time period for which the delegation will be actual. After the end of the period, the Iroha network removes the rights of sending transaction as behalf of initial user's account from the target user's account. |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0903] Inheritance of the account after period of inactivity |
| Status | DISCUSS - could be after MVP |
| Source | <ul style="list-style-type: none"> • Bakong (from Zilya Yagafarova) • Sora 2 project (from Zilya Yagafarova) |
| Preconditions | <ul style="list-style-type: none"> • The initial user has access to the target account in the Iroha network with permissions to configure inheritance • The descendant user has access to another account in the Iroha network with permissions to accept the inheritance |
| Use case flow | <ol style="list-style-type: none"> 1. The initial user prepares the instruction to configure inheritance rules 2. The initial user sends the transaction with the prepared instruction to the Iroha peer (<<include>> FR0100) 3. The Iroha peer responds with the result of the instruction to the initial user 4. Eventually, if the account of the initial user has no activity during the configured time period, the Iroha peer adds the key pair of the account of the descendant user to the list of signatories of the target account. 5. The descendant user receives rights to control the target account with its own key pair |
| Post-conditions | <ul style="list-style-type: none"> • The descendant user receives the control over the target account |
| Alternative flow | <ul style="list-style-type: none"> • On step 4, if the initial user perform any operation (instruction or query) the Iroha peers resets the inactivity timer. • On step 4, if the inheritance rules was configured to make several user as descendants, the Iroha peer will add all of these key pairs to the list of signatories for the target account. • TBD (what we should do if the account has more than one key pair attached?) |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0904] Distribution of fees according to business rules of the project |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Internal project (by Iurii Vinogradov) • Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> • The external system has access to the account with permissions to configure fees • The fee distribution business rules are defined by the external systems |
| Use case flow | <ol style="list-style-type: none"> 1. The external system configures the business rules of fee and distribution mechanism for the system (<<include>> FR0900) 2. Different users of the system performs operations, covered by the fee (<<include>> FR0901) 3. The Iroha network aggregates data about fee distribution by executing the DSL code, added to block storage on step 1. |

| | |
|------------------|---|
| | <ol style="list-style-type: none"> Regularly, each predefined interval of time the Iroha network applies distribution of fees by executing code, added to block storage on step 1. The external system eventually requests information about collected fees and corresponding assets distribution during defined period (<<include>> FR0200) |
| Post-conditions | <ul style="list-style-type: none"> All fees are collected and distributed according to the business rules The external system has access to the information about distribution of assets from collected fees |
| Alternative flow | <ul style="list-style-type: none"> On step 4, the distribution mechanism can be triggered manually by the external system On step 5, the request can be done in keep-alive mode, using FR0208 |
| Exception flow | N/A |

| | |
|-----------------------|---|
| Use case title | [FR0905] Managing the list of signatories |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> The user A has access to the account A in Iroha network with permissions to change its own list of signatories The user B has access to the account B in the Iroha network with permissions to be added as a signatory |
| Use case flow | <ol style="list-style-type: none"> The user A sends the transaction with instruction to add account of the user B to the list of signatories of the account A (<<include>> FR0100) <ol style="list-style-type: none"> This step is required to avoid flooding of list of pending transactions for account B in case in many users will add this account as a signatory without confirmation. The user B requests the list of pending requests to be added as a signatory (<<include>> FR0200) The user B sends the transaction with confirmation instruction for assigning account B to the list of signatories of account A (<<include>> FR0100) The Iroha network adds key pair of account B to the list of signatories of account A The user B receives possibility to sign the transaction on behalf of account A |
| Post-conditions | <ul style="list-style-type: none"> The user B has possibility to sign transactions on behalf of the account of user A |
| Alternative flow | <ul style="list-style-type: none"> On step 2, if the user B granted permission to user A to add account B as a signatory, the Iroha network will perform the operation without requesting confirmation from the user B. |
| Exception flow | <ul style="list-style-type: none"> On step 3, if the user B did not confirmed the operation within the timeout (defined in the network configuration), the initial request created on step 1 will be dismissed. Use case flow stops. |

| | |
|-----------------------|---|
| Use case title | [FR0906] Making decisions by parliament voting |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project – not required in the first version |
| Preconditions | <ul style="list-style-type: none"> There are N user in the system, which accounts are selected to be a member of parliament All of these accounts are added as a signatory to the system central parliament account The system central parliament account has quorum which is required for making decisions by majority of parliament members The initiator of the decision has permissions to make the initial transaction with the proposal |
| | |

| | |
|------------------|---|
| Use case flow | <ol style="list-style-type: none"> 1. The initiator of decision creates a transaction with the proposal body <ol style="list-style-type: none"> a. If the decision influences internal mechanics of the Iroha network, it should be represented in form of sequence of instructions, written using DSL b. If the decision influences external entities, it should be added as a document to the transaction payload 2. The initiator of the decision sends the transaction to the network (<<include>> FR0100) 3. All parliament members can get the list of pending transactions using their key pairs as a filter to get all pending decisions (<<include>> FR0203) 4. All parliament members makes their decision and sends corresponding transaction to the network 5. In case if majority of decisions reached, the Iroha network applies decision to the block store <ol style="list-style-type: none"> a. For sequence of instructions, they becomes executed and applied to the block store b. For the document, it becomes available as a immutable payload of the network |
| Post-conditions | <ul style="list-style-type: none"> • The decision approved and saved forever in the block storage |
| Alternative flow | <ul style="list-style-type: none"> • On step 5, if the majority of votes cannot be reached because of too much "contra" votes, the decision becomes rejected. • On step 5, if the majority of votes cannot be reached because of timeout of voting, the decision becomes rejected. |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0907] Management of non-fungible assets |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Internal project by Iurii Vinogradov – not required on the first release |
| Preconditions | <ul style="list-style-type: none"> • The user need to manage assets, which has unique characteristics for each entity |
| Use case flow | <ol style="list-style-type: none"> 1. The user receives request to store and manipulate virtual assets with uniques characteristics at each entity <ol style="list-style-type: none"> a. It can be virtual representations of unique real objects, like cars, houses, arts, etc. b. It can be pure virtual representation of unique entities which can be summed up into the single class 2. The user creates the asset type by sending corresponding ISI in the transaction (<<include>> FR 0100) to the Iroha peer 3. The user configures types of characteristics for the created asset type by sending corresponding ISI in the transaction to the Iroha peer 4. The user configures permissions for all related entities in the Iroha network so they can manage created type of assets; the user Sends corresponding ISI in the transaction to the Iroha peer 5. The user performs operations with assets of created asset type <ol style="list-style-type: none"> a. Creates new assets b. Changes characteristics of each asset |
| Post-conditions | <ul style="list-style-type: none"> • There is a new type of non-fungible assets in the Iroha with predefined list of unique characteristics |
| Alternative flow | <ul style="list-style-type: none"> • On step 3, characteristics can be mutable and immutable. Hence, on step 5, if all characteristics are immutable, there is no way to change them after minting new assets. |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0908] Nominating validator by staking assets |
| Status | DISCUSS - could be after MVP |
| Source | <ul style="list-style-type: none"> • Sora 2 project |
| Preconditi | |

| | |
|------------------|--|
| ons | <ul style="list-style-type: none"> There is a user of the network, which wants to take a validator role by staking needed amount of tokens as a guarantee of the honesty and responsibility. Let's call that user "the validator candidate" The validator candidate has running machine with Iroha node prepared for joining the Iroha network and become validating node. <p>TBD, need requirements details from the Sora project Pavel Golovkin</p> |
| Use case flow | <ol style="list-style-type: none"> The validator candidate sends predefined amount of assets to the special account for staking them in wish to become a nominated validator <p>TBD, need requirements details from the Sora project Pavel Golovkin</p> |
| Post-conditions | <ul style="list-style-type: none"> The validator was nominated and selected successfully; the machine of validator joined as a validation node to the Iroha. |
| Alternative flow | TBD |
| Exception flow | TBD |

| | |
|-----------------------|--|
| Use case title | [FR0909] Slashing of stakes for inappropriate behaviour of validator |
| Status | DISCUSS - could be after MVP |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> There is a validator in the system which performed staking of assets and was nominated by the voting to take a validator role (see FR0908) Validator's node is connected to the Iroha network and works with not stable results |
| Use case flow | <ol style="list-style-type: none"> The government members in the project get information about the "bad behaviour" of the validator The government member creates the proposal for slashing of validator's stake in terms of complex instruction written on DSL and sends it to the Iroha network (<<include>> FR0104) The government members votes for the proposal (<<include FR0906) In case if proposal gets majority of votes, the Iroha network executes slashing operation applies results to the storage |
| Post-conditions | <ul style="list-style-type: none"> The validator's stake was "slashed" in the favour of the overall budget of the "governance" structure |
| Alternative flow | <ul style="list-style-type: none"> On step 2, the decision about bad behavior can be made automatically if there is a possibility to operate with meta-parameters of the Iroha network from the DSL. Hence on step 3, the slashing proposal may be generated also automatically. |
| Exception flow | N/A |

| | |
|-----------------------|--|
| Use case title | [FR0910] Obtaining a list of trusted peers |
| Status | DISCUSS - could be after MVP |
| Source | <ul style="list-style-type: none"> Internal project (Iurii Vinogradov) |
| Preconditions | <ul style="list-style-type: none"> There are more than one peer in the current Iroha network The user has access to the account with permission for receiving information about trusted peers |
| Use case flow | <ol style="list-style-type: none"> The user sends the query with request for list of trusted peers to the Iroha peer (<<include>> FR0100) The Iroha peer returns result of the query to the user The user can check the validity of the query by the Merkle proof attached to the result |

| | |
|------------------|--|
| Post-conditions | <ul style="list-style-type: none"> The user has the list of the peers attached to the network which can be used for further requests. |
| Alternative flow | N/A |
| Exception flow | N/A |

| | |
|------------------|---|
| Use case title | [FR0911] Configuring a minimum limit of assets needed to create an account |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> There is a working Iroha network in normal conditions There is an administrator who has access to the root account with all required permissions |
| Use case flow | <ol style="list-style-type: none"> The administrator prepares a rule description which will disallow the creation of new account without transferring configurable amount of assets on it at the same time The administrator sends the rule to the Iroha network via transaction to the Iroha peer (<<include>> FR0100) (TBD) <ul style="list-style-type: none"> - we need to define how it will be implemented - by internal configuration or by the free-configurable trigger with conditions) The Iroha peer returns result of the transaction execution |
| Post-conditions | <ul style="list-style-type: none"> For creation of new accounts the Iroha network will require configured amount of tokens transferred to that account simultaneously |
| Alternative flow | <ul style="list-style-type: none"> On step 1, the rules for account creation may be different (depending on the implementation approach): list of assets may include more than single asset, other operations may be required for the account to be created. (TBD) |
| Exception flow | N/A |

| | |
|-----------------|--|
| Use case title | [FR0912] Creation of account with configured minimum amount of tokens |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Preconditions | <ul style="list-style-type: none"> There is a working Iroha network in normal conditions There is a configured rule for the account creation prerequisites (<<include>> FR0911) There is a user manager which has access to the corresponding account which has permissions to create new users and transfer tokens There is a new user who has a intent to create a new account in the network and has a confirmation of paying |
| Use case flow | <ol style="list-style-type: none"> The users sends the request to the user manager together with the confirmation of making payment for getting minimum amount of tokens which are required by the rule for account creation <ol style="list-style-type: none"> For Sora 2, it will be a transfer from one of connected blockchains via the bridge. (Pavel Golovkin – needs clarification from your side) The user manager initiates the account creation by preparing the transaction (which will include at least two instructions: account creation and transfer of requested amount of tokens to that account) and sending the transaction to the Iroha peer (<<include>> FR0100) The Iroha peer get the transaction, validates the amount of tokens required to create the account and returns result of the operation to the user manager. The user manager returns the status of operation to the user. |
| Post-conditions | <ul style="list-style-type: none"> The account for the new user is created, the balance corresponds to the amount of assets in the confirmation provided on step 1. |

| | |
|------------------|---|
| Alternative flow | <ul style="list-style-type: none"> On step 1, the way of confirmation may differ from project to project (TBD) On step 2, the account creation procedure may run automatically by fired trigger if the amount of required tokens can be minted automatically, too. |
| Exception flow | N/A |

Non-functional requirements

Non-functional requirements (also named as "Quality attributes") describes the behavior of the system not directly related to the functions of the system, but answers the question "How system works?". The template for all quality attributes should follow the example ([link to source](#)):

| Quality attribute name | [NFR0000] Example quality attribute; ID should be unique |
|------------------------|---|
| Status | <div>DISCUSS</div> <div>DECIDED</div> <div>POSTPONE</div> |
| Source | Source (or list of sources) of the quality characteristic: project name / stakeholder name / document title (e.g., whitepaper) / etc. |
| Source of stimulus | Entity, which initiates the stimulus, may be one of system users, another software system, etc. |
| Stimulus | Condition, which requires the response from the system |
| Environment | Definition of specific conditions when stimulus occurs, and which is important for the result of response. Typical environment values are: normal operation, overload of requests, starting up the system, etc. |
| Artifact | Particular subject of stimulus, may be the whole system, some subset of parts or single part of the system. |
| Response | Result of reaction of the system to the stimulus |
| Response measure | Measurable characteristic of the response, which can be checked for understanding how the system satisfies the requirements |



Quality attributes system standard

The systematization of quality attributes should follow the approach in standard [ISO/IEC 25010:2011](#)

00. Performance

| Quality attribute name | [NFR0001] Transaction processing speed |
|------------------------|---|
| Status | <div>DISCUSS</div> |
| Source | <ul style="list-style-type: none"> Iroha 2 white-paper Sora 2 project |
| Source of stimulus | Client applications of the Iroha network |
| Stimulus | Client application sends transactions to the Iroha network |
| Environment | Normal operation of the system |
| Artifact | Whole Iroha network |
| Response | The Iroha peer accepts the transactions and adds them to the blockchain |
| Response measure | <p>The Iroha network should process at least 20.000 transactions per second</p> <p>For Sora 2 project:</p> |

| | |
|--|--|
| | <ul style="list-style-type: none"> • normal load: 6 transactions per second (TPS) • heavy load: 50 TPS • peak load: 1000 TPS <p>By Makoto Takemiya</p> <ul style="list-style-type: none"> • target throughput: 20.000 TPS (in the aim to beat competitors) |
|--|--|

| Quality attribute name | [NFR0001] Delay of block creation |
|------------------------|---|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Iroha 2 white-paper • Sora 2 project |
| Source of stimulus | Client applications of the Iroha network |
| Stimulus | Client application sends transactions to the Iroha network |
| Environment | Normal operation of the system |
| Artifact | Whole Iroha network |
| Response | The Iroha peer accepts the transactions and to the block |
| Response measure | The Iroha network should create new block each 2-3 seconds |

| Quality attribute name | [NFR0002] Delay of restarting the peer |
|------------------------|---|
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Iroha 2 white-paper • Bakong project (Zilya Yagafarova) • Sora 2 project (Pavel Golovkin) |
| Source of stimulus | Administrator of the host with running Iroha peer |
| Stimulus | Administrator restarts the Iroha peer (manually or automatically by external script) |
| Environment | Normal operation of the system; the block storage is not corrupted. |
| Artifact | Current Iroha peer |
| Response | <p>The Iroha peer restarts and restores the WSV in the storage using one of two modes:</p> <ul style="list-style-type: none"> • the fastInit mode, when all transactions are applied without the verification of the block storage consistency • the strictInit mode, when all transactions and blocks are verified to have correct signature and follow the business rules |
| Response measure | <p>The Iroha peer successfully restarted, with following metrics:</p> <ul style="list-style-type: none"> • for fastInit mode: <ul style="list-style-type: none"> ◦ restart should take not more than 50 (??) ms per block TBD • for strictInit mode: <ul style="list-style-type: none"> ◦ restart should take not more than 200 (??) ms per block TBD |

| Quality attribute name | [NFR0003] Performing as expected on predefined hardware |
|------------------------|---|
| Status | DISCUSS |
| | |

| | |
|--------------------|--|
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Source of stimulus | The Validator |
| Stimulus | Attempt to run Iroha peer on the validators' machines |
| Environment | Normal operation of the system; the host machine is satisfying minimal requirements from the Iroha documentation |
| Artifact | The Iroha peer |
| Response | The Iroha peer start the execution |
| Response measure | The Iroha peer should start working properly, without functional issue and satisfying all non-functional requirements to performance |

| | |
|-------------------------------|--|
| Quality attribute name | [NFR0004] Providing enough capacity for user's accounts |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Makoto Takemiya |
| Source of stimulus | Users of the network |
| Stimulus | Creation of personal accounts in the network |
| Environment | Normal operation of the system |
| Artifact | The Iroha network and block storage |
| Response | The Iroha network allows to register as much users as needed in the target system; block storage can successfully keep all the relevant data |
| Response measure | The Iroha network can successfully handle at least 20 million of users' accounts |

01. Portability

| | |
|-------------------------------|--|
| Quality attribute name | [NFR0100] Easy integration from client side applications |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Iroha 2 white-paper Sora 2 project |
| Source of stimulus | Client applications of the Iroha network |
| Stimulus | Client application needs interaction with the Iroha network |
| Environment | Development of the client-side applications |
| Artifact | Client-side applications |
| Response | There are client libraries with efficient SDK available. |
| Response measure | Client libraries available for following programming languages and platforms: <ol style="list-style-type: none"> iOS (Swift) Android (JVM compatible: Java or Kotlin) Electron desktop (JavaScript or TypeScript) |

| | |
|-------------------------------|--|
| Quality attribute name | [NFR0101] Horizontal scalability of the network size |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |

| | |
|--------------------|--|
| Source of stimulus | User of the Iroha network with permissions to change list of validating peers |
| Stimulus | Sending operation of adding more validating peers to perform horizontal scalability of the network |
| Environment | Normal system functioning |
| Artifact | Whole Iroha network |
| Response | The size of Iroha network was increased by new validating peers, provided in the request |
| Response measure | The size of Iroha network should be increased at least up to 22 validating peers TBD |

| | |
|------------------------|---|
| Quality attribute name | [NFR0102] Adaptability for different environments and projects |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Source of stimulus | The operations engineer with permissions to configure the network |
| Stimulus | Changing system configuration parameters |
| Environment | On system start |
| Artifact | Whole Iroha network |
| Response | The system configuration changes according to the request from the user |
| Response measure | <p>The system configuration should provide possibility to tune up following parameters:</p> <ul style="list-style-type: none"> block generation time limit; block transaction count limit; expiration time for pending MST transactions; consensus mechanism parameters; etc. <p>TBD - we need to define metrics there</p> |

| | |
|------------------------|---|
| Quality attribute name | [NFR0102] Flexibility of integrated DSL for complex operations and triggers |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project Internal project (Iurii Vinogradov) |
| Source of stimulus | The system engineer of the external project |
| Stimulus | Need to describe the logic of core operations within the Iroha network |
| Environment | On design and implementation of the project over Iroha |
| Artifact | DSL of the Iroha |
| Response | The DSL allows to describe all required manipulations with the internal entities in the Iroha network |
| Response measure | The DSL provides system of data-manipulation rules, so it can be used to manipulate all entities in the Iroha network (available by permissions) and to manage control flow by using functions, conditions, loops, etc. |

| | |
|------------------------|--|
| Quality attribute name | [NFR0103] Reusability of the Iroha interface during integration with external systems |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Internal project (Iurii Vinogradov) |

| | |
|--------------------|--|
| Source of stimulus | The external software system |
| Stimulus | Performing operations with the Iroha network using standardized interfaces |
| Environment | Normally functioning Iroha network |
| Artifact | API of the Iroha network |
| Response | The Iroha network should provide the convenient interface for interaction from the other systems. |
| Response measure | The interface should be widely used and correspond to the industrial standard. Good candidate for such interface is HTTP(S) for single requests and WebSockets for continuous communication. |

| | |
|------------------------|---|
| Quality attribute name | [NFR0103] Configurability of permission |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Internal project (Iurii Vinogradov) |
| Source of stimulus | The external client of the Iroha network |
| Stimulus | Changing permissions for some entities in the Iroha network: <ul style="list-style-type: none"> accounts triggers groups of accounts global permissions |
| Environment | Normally running Iroha network |
| Artifact | Permission of entities in the Iroha network |
| Response | The Iroha provides flexibility in the permissions configuration, so the user may configure it for each mentioned entity |
| Response measure | TBD – define the measure in concrete details |

02. Security

| | |
|------------------------|---|
| Quality attribute name | [NFR0200] Non-repudiation of data between peer and client |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Sora 2 project |
| Source of stimulus | Client-side applications of the Iroha network |
| Stimulus | Client-side application sends the request to the Iroha peer and gets the response |
| Environment | Normal functioning system |
| Artifact | Connection between client-side application and Iroha peer. |
| Response | Client-side application checks the authenticity of the response |
| Response measure | Client-side application can be sure that data received from the Iroha peer is not changed by the man-in-the-middle attack |

03. Usability

| | |
|------------------------|---|
| Quality attribute name | [NFR0300] Convenient documentation for different user types |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> Makoto Takemiya Generic approach to widely used software solutions |

| | |
|--------------------|---|
| Source of stimulus | Different types of users of the Iroha <ul style="list-style-type: none"> • Executives of commercial projects • Software engineers, who works on the commercial software on the top of Iroha • Software engineers, who wants to contribute to Iroha development • Students and researchers of blockchain solutions |
| Stimulus | Need to get all related information about the Iroha |
| Environment | In process of research and development of digital solutions |
| Artifact | Documentation of the Iroha |
| Response | Documentation can provide excessive information about all entities and fundamentals of Iroha |
| Response measure | Each user of all different types can explore the documentation and get answer on required question within acceptable amount of time. |

04. Reliability

| | |
|------------------------|--|
| Quality attribute name | [NFR0400] Available proofs of efficiency of technical decisions and implementation |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Makoto Takemiya |
| Source of stimulus | Analysts of blockchain solutions |
| Stimulus | Request to get excessive information about proofs for efficiency of technical decisions and implementation design |
| Environment | In process of analyzing effectiveness and soundness of the Iroha |
| Artifact | Documentation of the Iroha |
| Response | Documentation provides information about experiments, benchmarks and researches made for making all major decisions and designing structure of the software solution for the Iroha |
| Response measure | All provided data is clear for understanding of specialists and can be easily verified by repeating the same experiments or benchmarks mentioned as proofs |

| | |
|------------------------|--|
| Quality attribute name | [NFR0401] Safety of the integrated DSL language for triggers |
| Status | DISCUSS |
| Source | <ul style="list-style-type: none"> • Internal project (Iurii Vinogradov) • Iroha 2 whitepaper by Makoto Takemiya |
| Source of stimulus | Code in triggers or smart contracts, defined on the integrated DSL language |
| Stimulus | Execution of the smart contract or firing of the trigger |
| Environment | Running Iroha network in normal mode |
| Artifact | Stability of all peers in the network |
| Response | The DSL should be designed to prevent the possibility of crashing whole network or it's parts by executing the code |
| Response measure | The DSL should prevent any kind of attacks, including: <ul style="list-style-type: none"> • buffer overflow • infinite loop • uncaught exceptions raising • etc. TBD |

Questions

Below is a list of questions to be addressed as a result of this requirements document:

| Question | Outcome |
|----------|---------|
| | |

Not Doing