#

Service

NOTE: Commands shown in this document are for illustration purpose only. For accurate syntax of commands, please refer tocli docs .

#

Summary

IRIS Service (a.k.a. iService) is intended to bridge the gap between the blockchain world and the conventional application world. It formalizes off-chain service definition and binding (provider registration), facilitates invocation and interaction with those services, and mediates service governance process (profiling and dispute resolution).

#

Service Definition

#

Service interface schema

Any user can define services on the blockchain. The interface of a service must be specified in terms of itsinput andoutput using the standard language ofJSON Schemaopen in new window. Here is an example:

{ "input" : { "schema" : "http://json-schema.org/draft-04/schema#" , "title" : "service-def-input-body-example" , "description" : "Schema for a service input body example" , "type" : "object" , "properties" : { "base" : { "description" : "base token denom" , "type" : "string" } , "quote" : { "description" : "quote token denom" , "type" : "string" } } , "required" : ["base" , "quote"] } , "output" : { "schema" : "http://json-schema.org/draft-04/schema#" , "title" : "service-def-output-body-example" , "description" : "Schema for a service output body example" , "type" : "object" , "properties" : { "price" : { "description" : "price" , "type" : "number" } } , "required" : ["price"] } }

#

Commands

create a new service definition

iris txservice define--name = < service-name> < schemas-json or path/to/schemas.json> --description = < service-description> --author-description= < author-description> --tags = < tag1,tag2,.. .> --from = < key-name> --chain-id= < chain-id>

query service definition

iris qservice definition< service-name>

#

Service Binding

Whoever is willing to provide a service as specified by an existing definition can do so by creating abinding to that definition. A binding essentially consists of four components:provider address ,pricing ,deposit andquality of service .

#

Provider address

A provider address represents anendpoint that aservice provider (i.e., an off-chain server/daemon) listens to for incoming requests. Before a provider can accept and process any service requests, its human operator/owner has to create a blockchain address for it and execute abind transaction to attach this provider address to the service definition in question.

To invoke a service, a user/consumer sends a request transaction targeting the provider address of a valid binding; the provider detects and processes the incoming request, and sends back the result in a response transaction.

#

Pricing

The pricing object specifies how the provider charges for its service; it must conform to this schema , and the following is a valid instance:

{ "price" : "0.1iris" , "promotions_by_time" : [{ "start_time" : "2020-01-01T00:00:00Z" , "end_time" : "2020-03-31T23:59:59Z" , "discount" : 0.7 } , { "start_time" : "2020-04-01T00:00:00Z" , "end_time" : "2020-06-30T23:59:59Z" , "discount" : 0.9 }] } Providers can opt to accept tokens other thaniris for their services, e.g.,0.03link . Price is a factor that consumers will consider when picking from a number of providers that offer the same service.

#

Deposit

Operating a service provider signifies serious responsibility, therefore, a deposit is required for creating a binding. The deposit amount must be larger than thedeposit threshold , derived asmax(MinDepositMultiple * price, MinDeposit) . If a provider fails to respond to a request before it times out, a small portion of its binding deposit, i.e.,SlashFraction * deposit , will be forfeited and destroyed. Should the deposit drop below the threshold, the binding would be disabled temporarily until its owner re-activates it by adding more deposit.

NOTE: service/MinDepositMultiple ,service/MinDeposit andservice/SlashFraction are system parameters that can be changed through on-chaingovernance .

#

Quality of service

This quality commitment is declared in terms of average number of blocks it takes the provider to send a response back to the blockchain. It is another factor that consumers take into account when choosing potential providers.

#

Commands

Service bindings can be updated at any time by their owners to adjust pricing, increase deposit or change QoS commitment; they can be disabled and re-enabled as well. If a provider owner no longer wants to offer the service, she needs to disable the binding and wait for a certain period of time before she can claim back her deposit.

create a new service binding

iris txservice bind < service-name> < provider-address> --deposit = < deposit> --qos = < qos> --pricing = < pricing-json or path/to/pricing.json> --from = < key-name> --chain-id= < chain-id>

update a service binding

iris txservice update-binding< service-name> < provider-address> --deposit = < added-deposit> --qos = < qos> --pricing = < pricing-json or path/to/pricing.json> --from = < key-name> --chain-id= < chain-id>

enable an inactive service binding

iris txservice enable < service-name> < provider-address> --deposit = < added-deposit> --from = < key-name> --chain-id= < chain-id>

disable an active service binding

iris txservice disable< service-name> < provider-address> --from = < key-name> --chain-id= < chain-id>

request refund of service binding deposit

iris txservice refund-deposit< service-name> < provider-address> --from = < key-name> --chain-id= < chain-id>

list all the bindings of a service

iris txservice bindings< service-name> --from = < key-name> --chain-id= < chain-id>

list all the bindings of a service, owned by a given account

iris qservice bindingsservice bindings< service-name> --owner < address>

query a specific service binding

iris qservice binding< service-name> < provider-address>

query the pricing schema

iris qservice schema pricing

#

Service Invocation

#

Request context

A consumer specifies how she would like to invoke a service by creating arequest context , which behaves like a smart contract that automatically generates the actual request(s). A request context consists of about a dozen parameters that can be roughly divided into four groups, as explained below.

#

Target and input

- service name
- : name of the target service to be called
- input data
- : json input that conforms to the input schema of the called service

#

Provider filtering

- provider list
- : comma separated addresses of candidate service providers
- service fee cap
- : maximum service fee the consumer is willing to pay for any call
- timeout
- : number of blocks the consumer is willing to wait for any response to come back

#

Response handling

- module
- : name of the module containing the callback function
- response threshold
- : minimum number of responses that must be received to invoke the callback

NOTE: These two parameters cannot be set from CLI or API; they are only available to other modules that use iService, such asoracle andrandom .

#

Repetition

- repeated
- : boolean flag indicating whether this request context is repeated or not
- frequency
- : number of blocks representing the frequency of repeated call batches
- total
- : total number of call batches, where a negative number means "unlimited"

#

Request batch

For a repeated request context,batches of new request objects will be generated at the specified frequency, until the total number of batches is reached or the consumer (i.e., context creator) runs out of fee. Only one request batch is generated for a non-repeated context.

A request batch is comprised of a number ofrequest objects, each representing a service call to a chosen provider; only those providers that charge a fee lower thanservice fee cap and commit to a QoS better thantimeout will be selected.

#

Commands

When a request context is successfully created, acontext id is returned to the consumer and the context is automatically started. The consumer can later update, pause and start the context at will; she can permanently kill the context as well.

create a repeated request context, with no callback

iris txservice call --service-name= < service name> --data = < request input> --providers = < provider list> --service-fee-cap= 1iris--timeout 50 --repeated --frequency = 50 --total = 100 --from = < key-name> --chain-id= < chain-id>

update an existing request context

iris txservice update< request-context-id> --frequency = 20 --total = 200 --from = < key-name> --chain-id= < chain-id>

pause a running request context

iris txservice pause< request-context-id> --from = < key-name> --chain-id= < chain-id>

start a paused request context

iris txservice start< request-context-id> --from = < key-name> --chain-id= < chain-id>

permanently kill a request context

iris txservice kill < request-context-id> --from = < key-name> --chain-id= < chain-id>

query a previously created request context by its id

iris qservice request-context< request-context-id>

list all the requests generated for a given request batch

iris qservice requests< request-context-id> < batch-counter>

list all the responses received for a given request batch

iris qservice responses< request-context-id> < batch-counter>

query a specific response

iris qservice response< request-id>

#

Service Response

A service provider monitors the blockchain, through either queries or event subscription, for any requests that are targeted toward itself. After processing such a request, the service provider sends back a response comprised of aresult object and an optional output object that conforms to the output schema of the service.

#

Service result schema

The result object must conform to thisschema , and here is an example of a valid instance:

{ "result" : { "code" : 400 , "message" : "user input out of range" } } The output object is required in the response only when the result code equals200 .

#

Commands

list all pending requests targeting a given provider

iris qservice requests< service-name> < provider>

query a specific request

iris qservice request< request-id>

send a response back, matching a specific request

iris txservice respond --request-id= < request-id> --result = '{"code":200,"message":"success"}' --data = < response output> --from = < key-name> --chain-id= < chain-id>

query the result schema

iris qservice schema result

#

Service Fees

Any user who creates service bindings and operates service providers should define awithdrawal address ; when the user withdraws service fees earned by her providers, this is where the fund will be sent to. If not set, the withdrawal address is the same as the user address.

#

Escrow

When a request object is generated, the associated service fee isnot paid to the targeted provider immediately; instead, the fee is kept in an internalescrow account for custody. When a response comes back in time (i.e., before the request times out), the corresponding fee (after tax) will be released from escrow to the provider; otherwise, the fee will be refunded to the consumer.

#

Tax

Right before a service fee is paid to a provider, atax , in the amount ofServiceFeeTax * fee is collected and sent to the community pool.

NOTE: service/ServiceFeeTax is a system parameter that can be changed through on-chaingovernance .

#

Commands

set withdrawal address

iris txservice set-withdraw-addr< withdrawal-address> --from = < key-name> --chain-id= < chain-id>

query withdrawal address of a given account

iris qservice withdraw-addr< address>

query a provider's earned fees

iris qservice fees< provider-address>

withdraw earned fees from all providers

iris txservice withdraw-fees--from = < key-name> --chain-id= < chain-id>

withdraw earned fees from a given provider

iris txservice withdraw-fees< provider-address> --from = < key-name> --chain-id= < chain-id>

#

Service Governance (TODO)