

KIP-103: Separation of Internal and External traffic


- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
 - [Broker Configuration](#)
 - [ZooKeeper](#)
 - [Protocol](#)
 - [Client](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)


Status

Current state: *Adopted*

Discussion thread: [here](#)

JIRA:

 Unable to render Jira issues macro, execution error. (0.10.2.0)

 Unable to render Jira issues macro, execution error. (0.11.0.0)

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

During the 0.9.0.0 release cycle, support for [multiple listeners](#) per broker was introduced. Each listener is associated with a security protocol, ip/host and port. When combined with the advertised listeners mechanism, there is a fair amount of flexibility with one limitation: at most one listener per security protocol in each of the two configs (`listeners` and `advertised.listeners`).

In some environments, one may want to differentiate between external clients, internal clients and replication traffic independently of the security protocol for cost, performance and security reasons. A few examples that illustrate this:

1. Replication traffic is assigned to a separate network interface so that it does not interfere with client traffic.
2. External traffic goes through a proxy/load-balancer (security, flexibility) while internal traffic hits the brokers directly (performance, cost).
3. Different security settings for external versus internal traffic even though the security protocol is the same (e.g. different set of enabled SASL mechanisms, authentication servers, different keystores, etc.)

As such, we propose that Kafka brokers should be able to define multiple listeners for the same security protocol for binding (i.e. `listeners`) and sharing (i.e. `advertised.listeners`) so that internal, external and replication traffic can be separated if required.

Public Interfaces

Broker Configuration

A new broker config `listener.security.protocol.map` will be introduced so that we can map a listener name to a security protocol. The config value should be in the CSV Map format that is currently used by `max.connections.per.ip.overrides`. The config value should follow map semantics: each key should only appear once, but values may appear multiple times. For example, the config could be defined in the following way to match the existing behaviour:

```
listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
```

To ensure compatibility with existing configs, we propose the above as the default value for the new config.

The next step is to change the validation of `advertised.listeners` and `listeners` so that the listener name has to be one of the keys in `listener.security.protocol.map` (only security protocols are allowed currently). For example, the following would configure a broker with two different host:port pairs mapped to the same security protocol in two cases:

```
listener.security.protocol.map=CLIENT:SASL_PLAINTEXT,REPLICATION:PLAINTEXT,INTERNAL_PLAINTEXT:PLAINTEXT,INTERNAL_SASL:SASL_PLAINTEXT
advertised.listeners=CLIENT://cluster1.foo.com:9092,REPLICATION://broker1.replication.local:9093,INTERNAL_PLAINTEXT://broker1.local:9094,INTERNAL_SASL://broker1.local:9095
listeners=CLIENT://192.1.1.8:9092,REPLICATION://10.1.1.5:9093,INTERNAL_PLAINTEXT://10.1.1.5:9094,INTERNAL_SASL://10.1.1.5:9095
```

We then introduce a second broker config as an alternative to `security.inter.broker.protocol`:

```
inter.broker.listener.name=REPLICATION
```

It is an error to set both `security.inter.broker.protocol` and `inter.broker.listener.name` at the same time. `inter.broker.listener.name` will be null by default, which means that the `PLAINTEXT` protocol will be used by default (as is currently the case).

Finally, we make it possible to provide different security (SSL and SASL) settings for each listener name by adding a normalised prefix (the listener name is lowercased) to the config name. For example, if we wanted to set a different keystore for the `CLIENT` listener, we would set a config with name `listener.name.client.ssl.keystore.location`. If the config for the listener name is not set, we will fallback to the generic config (i.e. `ssl.keystore.location`) for compatibility and convenience. For the SASL case, some configs are provided via a JAAS file, which consists of one or more entries. The broker currently looks for an entry named `KafkaServer`. We will extend this so that the broker first looks for an entry with a lowercased listener name followed by a dot as a prefix to the existing name. For the `CLIENT` listener example, the broker would first look for `client.KafkaServer` with a fallback to `KafkaServer`, if necessary.

ZooKeeper

Version 4 of the broker registration data stored in ZooKeeper will have listener names instead of security protocols in the elements of the `endpoints` array and an additional `listener.security.protocol.map` field. The latter is not strictly needed if we assume that all brokers have the same config, but it would make config updates trickier (e.g. two rolling bounces would be required to add a new mapping from listener name to security protocol). Also, we add an additional field instead of changing the `endpoints` schema to allow for rolling upgrades.

```
{
  "version": 4,
  "jmx_port": 9999,
  "timestamp": 2233345666,
  "host": "localhost",
  "port": 9092,
  "rack": "rack1",
  "listener_security_protocol_map": {
    "PLAINTEXT": "PLAINTEXT",
    "SSL": "SSL",
    "SASL_PLAINTEXT": "SASL_PLAINTEXT",
    "SASL_SSL": "SASL_SSL"
  },
  "endpoints": [
    "CLIENT://cluster1.foo.com:9092",
    "REPLICATION: //broker1.replication.local:9093",
    "INTERNAL_PLAINTEXT: //broker1.local:9094",
    "INTERNAL_SASL://broker1.local:9095"
  ]
}
```

Protocol

Version 3 of `UpdateMetadataRequest` will be introduced and the elements of the `end_points` array would also have a `listener_name` field.

```
UpdateMetadata Request (Version: 3) => controller_id controller_epoch [partition_states] [live_brokers]
  controller_id => INT32
  controller_epoch => INT32
  partition_states => topic partition controller_epoch leader leader_epoch [isr] zk_version [replicas]
    topic => STRING
    partition => INT32
    controller_epoch => INT32
    leader => INT32
    leader_epoch => INT32
    isr => INT32
    zk_version => INT32
    replicas => INT32
  live_brokers => id [end_points]
    id => INT32
    end_points => port host listener_name (new) security_protocol_type
      port => INT32
      host => STRING
      listener_name => String (new)
      security_protocol_type => INT16
```

Client

Listener names only exist in the brokers, clients never see them.

Proposed Changes

We would have to change a number of places in the code that currently use `SecurityProtocol` as a key to use the listener name instead. A few examples:

1. Acceptor thread
2. Metadata request handler
3. `ReplicaManager`
4. Broker class

The changes are mostly mechanical and don't affect public API.

We would also have to change the various authenticator classes to look for security configs for the relevant listener name before falling back to the generic ones.

As stated previously, clients never see listener names and will make metadata requests exactly as before. The difference is that the list of endpoints they get back is restricted to the listener name of the endpoint where they made the request. In the example above, let's assume that all brokers are configured similarly and that a client sends a metadata request to `cluster1.foo.com:9092` and it reaches broker1's `192.1.1.8:9092` interface via a load balancer. The security protocol would be `SASL_PLAINTEXT` and the metadata response would contain `host=cluster1.foo.com,port=9092` for each broker returned.

The exception is ZooKeeper-based consumers. These consumers retrieve the broker registration information directly from ZooKeeper and will choose the first listener with `PLAINTEXT` as the security protocol (the only security protocol they support).

Compatibility, Deprecation, and Migration Plan

As mentioned previously, the default value of `listener.security.protocol.map` maps the existing security protocols to a listener with the same name to maintain compatibility:

```
listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
```

For users upgrading, they should only use listener names once all the brokers have been upgraded to a version that supports listener names. ZooKeeper-based consumers will use the first listener with `PLAINTEXT` as the security protocol, so listener ordering is important in such cases.

Rejected Alternatives

1. Instead of adding the `listener.security.protocol.map` config, we could extend the protocol part of the listener definition to include both the listener name and security protocol. For example, `CLIENT+SASL_PLAINTEXT://192.1.1.8:9092`. This is appealing from a clarity perspective (the listeners are fully defined in a single config value), but it may lead to duplication between `listeners` and `advertised.listeners`. A way to avoid that issue (at the cost of loss of symmetry) would be for `advertised.listeners` to only include the listener name (we can infer the security protocol by looking at the `listeners` entry with the same name).

2. Assume that `listener.security.protocol.map` is the same in every broker. The slight benefit in terms of smaller broker registration JSON is not worth the additional operational complexity when it comes to changing the config values in a running cluster (two rolling upgrades would be needed in some simple cases).
3. Using hard-coded listener domains for internal and replication traffic. The config format is simpler and there's less scope for hard to understand configs. The main disadvantage is that it's a bit too specific and may need to be extended again as more sophisticated use cases appear. The current proposal is more general and it seems like a natural evolution of the existing system.