OMNI52 Istio Quick Reference

v1.30 – 2026.06

Architecture

Datapath & Components

Client → Ingress GW → Sidecar(src) → Sidecar(dst) → App. Hop ≈ 0.5–2 ms p99. mTLS handshake only on connect.
istiod: Pilot/Citadel/Galley combined, xDS push + CA.
istio-proxy: Envoy sidecar (mutating webhook) or ambient ztunnel.
Gateway: dedicated Envoys, Deployment + LB service.
Envoy version = Istio version + 8. Istio 1.30 → Envoy 1.38.

Install Profiles

default (prod), **minimal** (istiod only), **ambient** (sidecar-less), **remote** (MC workload cluster), **demo** (noisy). Canary via **--revision=1-30-1**.

```
istioctl install --set profile=default \
  --set values.global.proxy.resources.requests.cpu=100m
istioctl verify-install
```

Traffic Management

Gateway

Binds ports/hosts/TLS at the edge proxy. **servers[].port** + **tls.mode**: **SIMPLE**, **MUTUAL**, **PASSTHROUGH**, **ISTIO_MUTUAL**. **credentialName** points to a Secret in the same namespace as the gateway pod.

```
apiVersion: networking.istio.io/v1
kind: Gateway
metadata: {name: web-gw, namespace: istio-system}
spec:
  selector: {istio: ingressgateway}
  servers:
  - port: {number: 443, name: https, protocol: HTTPS}
    hosts: ["www.istio-quickref.de"]
    tls:
      mode: SIMPLE
      credentialName: web-cert
```

VirtualService

Routing rules. Binds **hosts** to **gateways** (or **mesh** for east-west). Match order is significant — the first matching rule wins. Header match case-insensitive, path match case-sensitive.

```
apiVersion: networking.istio.io/v1
kind: VirtualService
metadata: {name: reviews}
spec:
  hosts: [reviews]
  http:
  - match:
    - headers: {end-user: {exact: jason}}
    route:
    - destination: {host: reviews, subset: v2}
  - route:
    - destination: {host: reviews, subset: v1}
      weight: 90
    - destination: {host: reviews, subset: v3}
      weight: 10
    retries:
      attempts: 3
      perTryTimeout: 2s
      retryOn: gateway-error,connect-failure,refused-stream
    timeout: 10s
```

DestinationRule

Subsets (versions) + traffic policy (LB, connection pool, outlier detection, TLS). Takes effect only after VirtualService routing. **host** must be an FQDN or short name in the same namespace.

```
apiVersion: networking.istio.io/v1
kind: DestinationRule
metadata: {name: reviews}
spec:
  host: reviews
  trafficPolicy:
    connectionPool:
      tcp: {maxConnections: 100}
      http:
        http1MaxPendingRequests: 64
        http2MaxRequests: 1000
        maxRequestsPerConnection: 10
    outlierDetection:
      consecutive5xxErrors: 5
      interval: 30s
      baseEjectionTime: 60s
    loadBalancer:
      consistentHash:
        httpHeaderName: x-user-id
  subsets:
  - name: v1
    labels: {version: v1}
  - name: v2
    labels: {version: v2}
```

ServiceEntry

Pull external hosts into the mesh registry (DB, SaaS, REST APIs). **MESH_EXTERNAL** + **DNS** resolution is the clean default combo. Without a ServiceEntry, egress traffic stays a **BlackHoleCluster** when **outboundTrafficPolicy=REGISTRY_ONLY**.

```
apiVersion: networking.istio.io/v1
kind: ServiceEntry
metadata: {name: stripe-api}
spec:
  hosts: ["api.stripe.com"]
  ports:
  - {number: 443, name: https, protocol: HTTPS}
  resolution: DNS
  location: MESH_EXTERNAL
```

Sidecar

Limits what a workload sees of the mesh registry — critical for memory footprint and push latency in large clusters. Default: every sidecar gets config for all services. Mandatory above > 200 services!

```
apiVersion: networking.istio.io/v1
kind: Sidecar
metadata: {name: default, namespace: prod}
spec:
  egress:
  - hosts:
    - "./*"              # own namespace only
    - "istio-system/*"
    - "shared/*"
```

Security

PeerAuthentication

mTLS mode between sidecars. **STRICT**, **PERMISSIVE**, **DISABLE**. Scope: mesh (in **istio-system**), namespace, or workload via **selector**. Migrating to STRICT: first mesh-wide **PERMISSIVE**, then switch namespace by namespace.

```
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata: {name: default, namespace: prod}
spec:
  mtls: {mode: STRICT}
---
# Port exception: NLB health check without mTLS
spec:
  selector: {matchLabels: {app: legacy}}
  mtls: {mode: STRICT}
  portLevelMtls:
    8080: {mode: PERMISSIVE}
```

AuthorizationPolicy

L7 access control. **action**: **ALLOW** (default), **DENY** (precedence), **AUDIT**, **CUSTOM** (ext authz). Empty **rules**: blocks/allows everything (depending on action). Multiple policies combine AND within, OR across policies.

```
apiVersion: security.istio.io/v1
kind: AuthorizationPolicy
metadata: {name: reviews-allow, namespace: prod}
spec:
  selector: {matchLabels: {app: reviews}}
  action: ALLOW
  rules:
  - from:
    - source:
        principals:
        - cluster.local/ns/prod/sa/productpage
    to:
    - operation:
        methods: [GET]
        paths: ["/reviews/*"]
    when:
    - key: request.auth.claims[groups]
      values: ["reader", "admin"]
```

RequestAuthentication (JWT)

Validates the JWT, fills **request.auth.*** for the AuthorizationPolicy. Important: without an additional DENY policy with **notRequestPrincipals=["*"]**, unauthenticated requests can still get through.

```
apiVersion: security.istio.io/v1
kind: RequestAuthentication
metadata: {name: jwt, namespace: prod}
spec:
  selector: {matchLabels: {app: api}}
  jwtRules:
  - issuer: "https://auth.example.com"
    jwksUri: "https://auth.example.com/jwks"
    audiences: ["api.example.com"]
    forwardOriginalToken: true
```

Identity

SPIFFE URI **spiffe://<trust>/ns/<ns>/sa/<sa>**. Trust domain defaults to **cluster.local**; in multi-cluster it must be unique per cluster. Workload cert rotation every 24 h, automatic via Citadel.

OMNI52

Istio Quick Reference

v1.30 – 2026.06

Observability

Telemetry API

Metrics/logs/traces per workload or namespace.
Replaces `EnvoyFilter` telemetry mods and the old `values.telemetry.v2.*` settings.

```
apiVersion: telemetry.istio.io/v1
kind: Telemetry
metadata: {name: trace-prod, namespace: prod}
spec:
  tracing:
  - providers: [{name: tempo}]
    randomSamplingPercentage: 5.0
  metrics:
  - providers: [{name: prometheus}]
    overrides:
    - match: {metric: REQUEST_COUNT}
      tagOverrides:
        request_protocol: {operation: REMOVE}
  accessLogging:
  - providers: [{name: otel}]
    filter:
      expression: "response.code ≥ 400"
```

Metrics & Tracing

RED counter: `istio_requests_total`, latency: `istio_request_duration_milliseconds_bucket`. The `reporter` label = source/destination — in dashboards always aggregate on destination (otherwise double counting).
Istio propagates B3/W3C, it does not generate them. The app must re-set incoming trace headers (`x-request-id`, `x-b3-*`, `traceparent`) on the way out.

Performance & Tuning

Sidecar Sizing

Default 100m CPU / 128Mi RAM: not production-grade at high RPS.
Rule of thumb: 1 mCore per 100 RPS, +50 MiB per 1000 endpoints in the registry.
`Sidecar` resource: trims the registry → typically -70 % memory.

Pilot Tuning

`PILOT_PUSH_THROTTLE`: default 100
`PILOT_DEBOUNCE_AFTER`: default 100 ms
`PILOT_DEBOUNCE_MAX`: default 10 s
On push storms (many pod restarts) raise debounce; raise throttle for faster convergence above > 5000 workloads.

Ambient Mode (1.30 GA)

No more sidecar injection.
`ztunnel`: node DaemonSet, L4 mTLS.
`Waypoint Proxy`: optional, namespace-scoped, L7.
Opt-in: label `istio.io/dataplane-mode=ambient` per namespace.
Saves RAM at high pod counts, costs complexity when debugging.

Jobs & CronJobs (Sidecar Lifecycle)

App before proxy ready → connection refused. App done, sidecar still running → the job hangs.
Native Sidecar (K8s 1.29+ Beta, Istio 1.19+): proxy as an `initContainer` (`restartPolicy: Always`), proper lifecycle. Mesh-wide via istiod env `ENABLE_NATIVE_SIDECARS=true`.
Per pod (Istio 1.24+, takes precedence over the mesh flag): annotation `sidecar.istio.io/nativeSidecar: "true"` in the pod template (`"false"` forces the classic sidecar).
Pre-native: `holdApplicationUntilProxyStarts` against the start race, `trap` with `POST :15020/quitquitquit` on EXIT against shutdown hangs (fires on crash/signal too, not only on success).

```
# Native per pod (Istio 1.24+, beats mesh flag): pod template
metadata:
  annotations:
    sidecar.istio.io/nativeSidecar: "true"
---
# Pre-native fallback (without native sidecars):
metadata:
  annotations:
    proxy.istio.io/config: |                # against start
     race
      { "holdApplicationUntilProxyStarts": true }
spec:
  containers:
  - name: worker
    command: ["/bin/sh","-c"]
    args:
    - |
      trap 'curl -fsS -XPOST
     localhost:15020/quitquitquit||true' EXIT
      ./run-task
```

Graceful Drain: `EXIT_ON_ZERO_ACTIVE_CONNECTIONS`

On SIGTERM the sidecar only drains for `terminationDrainDuration` (default 5 s), then hard-exits — long-lived connections (gRPC streams, WebSockets, DB pools) are cut mid-flight (503/reset on rollout/scale-down).
Fix: `EXIT_ON_ZERO_ACTIVE_CONNECTIONS=true` (via `proxyMetadata` or `proxy.istio.io/config`) — pilot-agent polls active Envoy connections and stops the proxy as soon as they reach 0 instead of on a fixed timer.
Caveat: if a connection hangs (client never closes), the proxy blocks until `terminationGracePeriodSeconds` → SIGKILL. Raise the grace period accordingly. Mesh-wide via `meshConfig.defaultConfig.proxyMetadata`.

Multi-Cluster

Topologies & Setup

Primary-Remote: one istiod, several workload clusters.
Multi-Primary: istiod per cluster, shared root CA.
External Control-Plane: istiod on the outside.
Required: shared `trustDomain` (or `trustDomainAliases`), a `network` label per cluster, endpoint discovery via `istioctl create-remote-secret`.

```
istioctl create-remote-secret \
  --context=cluster-b \
  --name=cluster-b \
  | kubectl apply --context=cluster-a -f -
```

Diagnostics

First `istioctl` Tool During an Incident

`istioctl proxy-status` shows the sync state of every sidecar. `SYNCED` = config current, `STALE` = push in progress, `NOT SENT` = istiod sent nothing → check the Pilot logs.

```
istioctl proxy-status
istioctl proxy-config routes <pod>.<ns> -o json
istioctl proxy-config clusters <pod>.<ns>
istioctl proxy-config listeners <pod>.<ns>
istioctl proxy-config endpoints <pod>.<ns>
istioctl proxy-config secrets <pod>.<ns>

# Static lint: VirtualService/DR conflicts etc.
istioctl analyze -n prod

# Full bug report incl. config + logs (anonymised)
istioctl bug-report
```

Live Log of a Sidecar

`istioctl proxy-config log <pod> --level debug` sets the log level live without a pod restart. Component-specific, e.g. `--level rbac:debug,jwt:debug`. After diagnosis, set it back to `warning`.

Common Failure Modes

503 UC: upstream connect fail, app port does not match the service target.
503 NR: no route, the VirtualService does not match (wrong host match).
503 UF: upstream failure, TLS mismatch (PeerAuth `STRICT` vs. a client without a sidecar).
404: path match missed, check the order of the `http[]` rules.

`OUTPUT_CERTS` Scraping: `read: connection reset by peer`

App-originated mTLS (Prometheus/Alloy federation via `ISTIO_META_OUTPUT_CERTS` + `/etc/istio-certs/{cert-chain,key,root-cert}.pem`) breaks with an RST even though the target `PeerAuthentication` is `PERMISSIVE` and allow-all matches.
Almost always the cause: the client sidecar wraps the app mTLS in a second `ISTIO_MUTUAL` tunnel. A DR with `host: "*.local"` + `exportTo: ["*"]` (a typical STRICT migration helper in `istio-system`) matches direct pod-IP calls too via EndpointSlice lookup. Result: TLS-in-TLS. Target `15006` terminates only the outer layer; the inner TLS bytes land as plain text on the app port, the backend expects HTTP and closes.
PERMISSIVE is an inbound decision after transport termination – it heals nothing that happens outbound.
Fix: `excludeOutboundPorts` on the scraper pod (port-scoped) or a dedicated DR with `tls.mode: DISABLE` on the target service (more specific than `*.local`, so it wins).

```
# does the client sidecar wrap? alpn=istio* + sni=outbound_
    = proof:
istioctl pc cluster <pod>.<ns> --fqdn <target> -o json \
  | jq
    '.[].transport_socket.typed_config|{sni,alpn_protocols}'
# which *.local DR matches?
kubectl get dr -A -o json | jq -r '.items[]|
  select(.spec.host|test("\\*\\.local"))|
  .metadata.namespace+"/"+.metadata.name'
```

OMNI52

Istio Quick Reference

v1.30 – 2026.06

Gateway API (v1, kubernetes-sigs)

Status in 1.30

Istio implements Gateway API v1 (**Gateway**, **HTTPRoute**, **GRPCRoute**) as an equal alternative to **Gateway**/**VirtualService**. New projects: Gateway API. Existing setups: **networking.istio.io** stays supported, the two can coexist.

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata: {name: web, namespace: istio-system}
spec:
  gatewayClassName: istio
  listeners:
  - name: https
    hostname: www.istio-quickref.de
    port: 443
    protocol: HTTPS
    tls:
      certificateRefs:
      - {name: web-cert}
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata: {name: site, namespace: web}
spec:
  parentRefs: [{name: web, namespace: istio-system}]
  hostnames: ["www.istio-quickref.de"]
  rules:
  - matches: [{path: {type: PathPrefix, value: /}}]
    backendRefs: [{name: nginx, port: 80}]
```

Anti-Patterns

What You Should Not Do

Default sidecar resources in prod: OOM, push throttle.
Mesh VirtualService without a **Sidecar** resource: every sidecar gets every rule, push storm.
STRICT without migration: non-injected workloads break (jobs, external health checks).
EnvoyFilter as the default tool: reach for the Telemetry API, WasmPlugin, AuthorizationPolicy first. EnvoyFilter breaks between minors.
Tracing without app header propagation: spans tear apart.
Multi-cluster without a shared root CA: mTLS fails.
OUTPUT_CERTS without **excludeOutboundPorts**: the client sidecar wraps app mTLS twice (TLS-in-TLS) – **read: connection reset by peer** despite a PERMISSIVE PA.

From the OMNI52 Cheatsheet Factory

Related sheets: istio-cheatsheet.de (Istio in depth, German) · istio-spickzettel.de (German cheatsheet) · service-mesh-cheatsheet.de (Istio + Linkerd + Cilium) · kubernetes-cheatsheet.de (Kubernetes Core).

---

istio-quickref.de

Istio Quick Reference by OMNI52

Take it further, don't reinvent

Workshops & Architecture Reviews

OMNI52™ secures Kubernetes clusters for regulated enterprise environments through an automated service mesh built on Istio.

Architecture review, STRICT mTLS migration, AuthorizationPolicy hardening, Telemetry API pipeline. The output lands as a GitOps repo in your cluster: Helm charts, runbooks, diagnostic scripts. Your team keeps operating, not us.

Uli Renz, OMNI52 GmbH, Ebern · istio-consulting.de · free 30-min initial assessment