FortisX Whitepaper

Staking analytics and allocation layer for validator-based networks.

FortisX provides an analytics and policy layer on top of validators, staking pools, and custody setups. This whitepaper describes how the platform ingests on-chain data, evaluates risk, and supports transparent staking allocation decisions for institutional, operational, and infrastructure teams.

Introduction & scope

FortisX is a cloud platform for staking in validator-based networks. It combines validator and network analytics with a policy engine that manages how capital is allocated across validators and pools within each supported network. This whitepaper describes the technical foundations behind FortisX: how data is collected and processed, how risk is modeled, how allocation policies are evaluated, and how the platform is operated and exposed to external systems.

The document presents FortisX through its architecture, data model, and operational processes. Staking-related quantities such as native yields or rewards are treated as network-level parameters and observable metrics within the analytics and risk model. Throughout the text, “FortisX” denotes the analytics and policy engine layer that sits above validators, pools, and custody systems.

FortisX initially targets a set of leading validator networks such as Ethereum, Solana, Polkadot, Avalanche, and Cosmos, and is designed to extend to additional networks that follow similar principles.

Purpose of this document

The goals of the FortisX Whitepaper are:

- To provide a **transparent description** of how FortisX evaluates validator and network behaviour, models risk, and proposes staking allocations.
- To define a **shared vocabulary and data model** for networks, validators, pools, providers, alerts, policies, and allocation proposals.
- To explain the **operational and security discipline** behind the platform: monitoring, incident handling, auditability, and external reviews.
- To document the **interfaces**—especially the Analytics API—through which external systems can consume the same analytics that drive the internal policy engine.

The whitepaper is intended to be concrete enough for engineers, security teams, and risk committees to evaluate how FortisX works, while remaining independent of any particular custody setup or validator operator.

Intended audience

This document is written for several types of readers:

- **Infrastructure and devops teams** who need to understand how FortisX ingests, stores, and processes data, and how it integrates with existing validator or custody infrastructure.
- **Validator operators and staking providers** who want to see how their performance, configuration, and risk profile are evaluated.

- **Institutional clients and risk committees** who require a clear view of the models and policies behind allocation decisions, as well as the operational and security controls of the platform.
- **Integration partners and developers** who consume FortisX analytics via API, SDKs, or custom dashboards.

The whitepaper assumes familiarity with validator-based networks and basic staking concepts and is written so that protocol-level details can be followed without specialised expertise for each individual chain.

Scope and boundaries

The scope of this whitepaper is centred on the **analytics, risk, and policy** layer of FortisX. It covers:

- the architecture of the ingest pipeline, analytics services, risk scoring, and policy engine;
- the core data model and the main classes of metrics used for validators and networks;
- the operational model, reliability targets, security principles, and the role of external reviews;
- the Analytics API at a conceptual level and the way external systems interact with FortisX.

Related topics such as detailed protocol specifications, client-specific custody arrangements, legal and regulatory treatment, and tax or accounting considerations belong to the broader documentation set and to the internal frameworks of organisations that use FortisX.

Relationship to other documents

The FortisX Whitepaper is part of a broader documentation set:

- The **API documentation** provides detailed request/response schemas, error handling, and usage examples for the Analytics API.
- The **SDK and integration guides** focus on practical implementation steps for connecting FortisX to external systems.
- The **Legal notice & risks** chapter in this whitepaper summarises risk factors and the legal context relevant to validator-based networks and the use of FortisX.

Together, these materials are intended to give a complete and coherent picture: this whitepaper explains **why** the platform works the way it does; the API and SDK docs explain **how** to connect to it.

Structure of the whitepaper

The remainder of this document is organised as follows:

1. **Legal notice & risks** – formal disclaimers and a structured view of the main risk factors relevant to validator-based networks and to the FortisX platform.

2. **Platform overview** – a high-level view of FortisX, its role in the staking ecosystem, and the main product surfaces.

3. **Architecture & data flow** – the end-to-end data pipeline, core services, and non-functional requirements.

4. **Data model & core entities** – the objects and relationships that underpin all analytics and policies.

5. **Validator metrics** – the key measurements used to assess validator and pool behaviour.

6. **Network & decentralization metrics** – the indicators that describe network health, concentration, and decentralization.

7. **Risk modeling** – how metrics are transformed into factor scores and risk buckets.

8. **Policy engine & allocation rules** – how policies are defined, evaluated, and surfaced as allocation proposals.

9. **Operations & reliability** – how the platform is run, monitored, and continuously improved.

10. **Security, audits & compliance** – how FortisX is secured, reviewed, and made auditable for institutional users.

11. **Analytics API & roadmap** – external interfaces and a forward-looking view of the platform's evolution.

Readers who are primarily interested in integrating with FortisX may focus on the architecture, data model, and API sections. Those evaluating FortisX from a governance or risk perspective may pay particular attention to the metrics, risk modeling, policy engine, and security chapters.

Legal notice & risks

This whitepaper is a technical and descriptive document. It explains how the FortisX platform is designed, which data it relies on, and how its models and processes operate. It is not a commercial offer, not a marketing brochure, and not a contract.

The information provided here is intended to help engineers, operators, and risk or governance teams understand the architecture and assumptions behind FortisX. It does not replace professional advice, does not address the circumstances of any particular reader, and should not be used as the sole basis for financial, legal, tax, or regulatory decisions.

Use of FortisX, and participation in validator-based networks more generally, involves technical and economic uncertainty. Those risks exist independently of this document and independently of the FortisX platform.

Informational character of the document

The whitepaper describes how FortisX ingests data from validator-based networks, how it models and aggregates that data, how risk signals and policies are derived, and how the platform is operated and exposed through interfaces such as dashboards and APIs.

It does not prescribe how any reader should structure their own staking operations, risk appetite, or allocation strategies. Where examples are given, they are illustrative and cannot be assumed to fit a particular organisation, jurisdiction, or mandate.

Nothing in this text should be interpreted as:

- a recommendation to participate in any specific network, validator, or pool;
- an assurance about future network conditions, validator behaviour, or operational continuity;
- a statement that the use of FortisX can remove or neutralise underlying protocol or market risk.

Readers remain responsible for forming their own judgement and for obtaining independent professional advice where appropriate.

Risks related to validator-based networks

FortisX observes and analyses networks such as Ethereum, Solana, Polkadot, Avalanche, Cosmos, and similar validator-based systems. These networks are complex, evolving software and economic mechanisms. They are subject to a range of risks, including:

- design or implementation issues in consensus algorithms, clients, or smart contracts;
- governance and upgrade decisions that may change protocol rules or economic parameters;
- slashing and penalty regimes that affect validators and, indirectly, participants who delegate to them;

- changes in participation, liquidity, and market demand that influence the environment in which validators operate.

These risks are inherent to the networks themselves. FortisX can surface their effects in the form of metrics, analytics, and risk indicators, but cannot control or eliminate them.

Infrastructure and operational risks

Running validators and related infrastructure involves hardware, software, and operational complexity. FortisX does not operate validators as part of the platform described in this document; instead, it depends on a broader ecosystem of components and providers, such as:

- servers, data centres, networks, and cloud platforms;
- client software, indexers, and RPC endpoints;
- explorers and other data services that contribute context.

Failures, misconfigurations, and changes in behaviour at any of these layers may influence both the behaviour of validators and the quality or timeliness of the data that FortisX receives. The platform is built with monitoring, redundancy, and defensive design in mind, but cannot guarantee uninterrupted availability of all upstream or downstream systems.

Data quality and model limitations

Analytics and risk assessments in FortisX are based on a combination of on-chain observations, network metadata, provider information, and internally derived aggregates. As a result, there are unavoidable limitations:

- not all data is available at the same granularity or latency across all networks and providers;
- some signals may be incomplete, delayed, revised, or temporarily inconsistent;
- models and thresholds used to interpret data are subject to refinement as networks evolve and as more information becomes available.

The platform is designed so that these assumptions can be examined and updated. Nevertheless, any metric, aggregate, or risk indicator should be understood as the output of a model and a data pipeline, not as a definitive statement about the future. FortisX aims to make such outputs explainable and traceable, but does not claim that they are exhaustive or infallible.

Role and responsibilities of FortisX

FortisX is conceived as an analytics and policy layer that produces staking allocation and rebalancing decisions based on observable data and configurable rules. Unless otherwise agreed in separate arrangements:

- it does not hold private keys or custody client assets;
- it does not unilaterally execute staking operations on behalf of clients;
- it does not replace clients' own responsibilities for operational controls, compliance, or due diligence on networks, validators, or custodians.

Where FortisX output (such as analytics, alerts, or allocation proposals) is consumed by external systems, decisions based on that output remain under the control of the operators of those systems. They decide which signals to incorporate, which policies to adopt, and how strictly to enforce or automate them.

Regulatory and jurisdictional context

Regulation related to digital assets, staking, and infrastructure services varies by jurisdiction and is evolving. This whitepaper does not attempt to classify FortisX, any network, or any activity under a specific legal or regulatory framework.

Organisations that make use of FortisX, operate validators, or participate in staking remain responsible for:

- understanding the regulatory environment applicable to them;
- assessing whether and how FortisX can be used within their own policies and control frameworks;
- ensuring that their use of FortisX is consistent with any obligations they may have to clients, regulators, or other stakeholders.

FortisX may update this whitepaper over time to reflect material changes to the platform or its environment. Such updates are intended to improve transparency, but they do not change the fundamental point that risk cannot be outsourced to documentation.

Forward-looking aspects

Some passages of this whitepaper describe directions for future development of the platform, such as possible extensions to network coverage, metrics, models, or interfaces. These descriptions are included to provide context and should be read as indicative plans, not as binding commitments.

Actual implementation details and timelines may differ from what is described here, for reasons including technical constraints, security considerations, dependency on third-party infrastructure, and changes in the regulatory or market landscape.

In summary, this document explains how FortisX is designed to observe and organise information about validator-based networks and staking infrastructure operated by others. It does not remove uncertainty, and it does not transfer responsibility for decisions. The subsequent sections build on

this foundation by describing the platform’s architecture, data model, analytics, models, and operational discipline in more detail.

Platform overview

FortisX provides an analytics and policy layer for organisations that allocate capital into validator-based networks. Rather than operating validators or custodying assets, it focuses on making validator, pool, and network behaviour observable, assessable, and governable through metrics, risk indicators, and explicit allocation policies.

At a high level, FortisX observes how validators, pools, and networks behave; turns these observations into time-series metrics and risk indicators; and uses them to drive policies that define where capital can and cannot be allocated. The same analytics that support internal policies are exposed through dashboards and APIs so that operators and risk teams can inspect both the data and the reasoning behind allocation decisions.

FortisX is initially focused on major validator-based networks and is designed to extend to additional networks that expose sufficient data to support this analytic and policy layer.

##

Role in the staking ecosystem

Validator-based networks require several distinct layers to function in practice:

- protocols and clients that implement consensus and state transitions;
- validators and infrastructure providers that operate nodes and participate in consensus;
- custody and execution systems that hold keys and apply staking decisions;
- monitoring, analytics, and policy tooling that make behaviour observable and controlled.

FortisX focuses on the last of these layers. It does not define consensus, run the networks, or custody assets. Instead, it provides an analytics and policy plane: a consistent environment where validator and network behaviour can be monitored, where risk can be assessed, and where allocation and rebalancing decisions can be expressed as explicit policies.

The platform is designed to work alongside different operational setups. A single analytic and policy framework can be used across multiple validator providers, custodians, or internal staking implementations, so that allocation logic is not fragmented between tools or teams.

Core capabilities

Validator and network analytics

FortisX maintains a data pipeline that continuously ingests information from validator-based networks and related infrastructure. For each supported network, the platform focuses on:

- validator- and pool-level characteristics such as reliability, participation, penalties, configuration changes, and performance relative to peers;

- network-level indicators such as staking participation, concentration of stake, distribution of roles across providers, and changes in overall load and activity;
- observable patterns in delegator behaviour, including large changes in stake, shifts between pools, and emerging concentrations.

These observations are stored as time series and aggregates, making it possible to analyse both current conditions and how they evolve over time.

Risk modeling

On top of raw and aggregated metrics, FortisX applies a risk model. The model does not attempt to predict prices or protocol outcomes; instead, it organises information into a set of dimensions that are relevant for staking decisions, such as:

- technical reliability of validators and pools;
- concentration and decentralisation characteristics at the network and provider level;
- operational behaviour over time, including incident history and configuration changes;
- exposure to specific infrastructure or service providers.

Each dimension is expressed in terms of derived indicators and scores that can be inspected and revised as assumptions change. The purpose of these scores is to support transparent, repeatable decision-making about where capital may be allocated.

Policy engine and allocation proposals

FortisX includes a policy engine that turns analytics and risk signals into concrete allocation and rebalancing proposals. Policies are defined as explicit rules and constraints, for example:

- upper bounds on exposure to a single validator, pool, or provider;
- minimum decentralisation characteristics for a network to be eligible;
- exclusion of validators or pools that fall into designated high-risk buckets;
- limits on how quickly allocations may change in response to new data.

Given current allocations, observed metrics, and configured policies, the engine produces proposals for how capital could be distributed across validators and pools within each network. These proposals can be reviewed, approved, and executed through systems that hold and manage assets, while remaining traceable back to the data and rules that produced them.

Integration and execution model

FortisX is designed as a non-custodial component. It does not assume control over private keys or direct authority to move assets. Instead, it integrates with:

- validator and staking providers that implement the actual on-chain operations;
- custody platforms that manage keys and enforce internal governance;

- internal systems that record positions, limits, and approvals.

In a typical setup, FortisX produces analytics, alerts, and allocation proposals; external systems apply their own controls and approval processes; and, once actions are authorised, they can be executed in a way that is consistent with both FortisX policies and the organisation's internal requirements.

External interfaces

The same data and models that drive internal analytics and policies are exposed through:

- dashboards and views for validators, networks, and pools;
- alert streams that highlight large stake movements or sudden changes in risk indicators;
- an Analytics API and SDKs that allow external systems to query networks, validators, metrics, and risk assessments.

This ensures that FortisX does not act as an opaque "black box". Operators and external systems can inspect the inputs and reasoning that sit behind allocation decisions, and can build additional tooling on top of the same analytic foundation.

Design principles

The platform is guided by a small set of design principles:

- **Data first** – allocation and rebalancing decisions are derived from observable metrics and explicit policies, not from ad hoc judgement or opaque heuristics.
- **Separation of duties** – analytics, risk modeling, policy specification, and execution can be operated and audited separately, matching how institutional control frameworks are structured.
- **Network-agnostic core** – while details differ between Ethereum, Solana, Polkadot, Avalanche, Cosmos, and other networks, the analytic and policy framework is shared so that cross-network decisions can be made consistently.
- **Explainability and auditability** – metrics, scores, and policies are designed to be inspectable and reproducible over time, so that risk committees and operators can understand how specific decisions were reached.
- **Incremental evolution** – models, metrics, and network coverage are expected to evolve, but changes are made explicitly and tracked, rather than folded silently into the system.

These principles shape the more technical chapters that follow: the architecture and data flow, the data model and metrics, the risk model, the policy engine, and the operational and security practices around the platform.

Architecture & data flow

This section describes how FortisX is structured as a system and how data moves through it. The goal is not to specify particular implementation technologies, but to make clear which components exist, what each of them is responsible for, and how they interact to produce analytics, risk assessments, and allocation proposals.

At a high level, the platform is organised into three conceptual layers:

1. **Data acquisition** – ingesting and normalising information from validator-based networks and related infrastructure.
2. **Analytics and risk** – turning raw and aggregated data into metrics, indicators, and risk signals.
3. **Policies and interfaces** – applying configurable rules to produce allocation and rebalancing evaluations and proposals, and exposing data and proposals to external systems.

The same architecture is applied consistently across supported networks such as Ethereum, Solana, Polkadot, Avalanche, and Cosmos, while allowing for network-specific nuances in data sources and metrics.

Data sources

FortisX relies on multiple kinds of upstream information:

- **On-chain data** – blocks, validator sets, consensus participation, slashing and penalty events, staking transactions, rewards, and other protocol-level signals.
- **Validator and provider endpoints** – where available, telemetry or status endpoints exposed by validators, pools, or infrastructure providers.
- **Indexers and explorers** – services that pre-process on-chain data into higher-level events or aggregate views.
- **Network metadata** – protocol parameters, governance state, configuration changes, and release schedules.

These sources differ by network in terms of structure, latency, and availability. The architecture is designed to treat them as interchangeable inputs into a common ingest pipeline, rather than tying the platform to any single provider.

Ingest pipeline

The ingest layer is responsible for continuously collecting data from external sources, validating it, and transforming it into a format suitable for analytics. Conceptually, it is composed of:

- **Network-specific collectors** – processes dedicated to particular networks (for example, Ethereum, Solana, Polkadot, Avalanche, Cosmos) that understand the network's RPC and

indexing interfaces and know how to interpret network events.

- **Normalisation stage** – logic that converts heterogeneous data structures into FortisX's internal representation of networks, validators, pools, and events.
- **Validation and quality checks** – basic integrity checks such as:

- consistency of block heights and timestamps;

- detection of missing or out-of-order data;

- cross-checks between multiple providers, where available.

- **Ingest queues and buffering** – internal queues that decouple upstream variability from downstream processing, so that short-lived spikes or delays upstream do not immediately affect the rest of the system.

Ingest processes run continuously and incrementally. Rather than performing large batch imports, they track the current position in each data source (for example, block height or slot index) and advance as new data becomes available.

Storage and aggregation

Once data passes through the ingest pipeline, it is stored in a combination of time series and relational structures:

- **Raw event storage** – a log of low-level events (such as validator participation, penalties, configuration changes, and stake movements) that can be replayed if models need to be recomputed.
- **Time-series metrics** – regularly sampled measurements (per epoch, slot, block, or fixed interval) for validators, pools, and networks.
- **Aggregated views** – precomputed aggregates used in analytics and risk modeling, such as:

- rolling averages and percentiles of validator performance;

- stake concentration across providers or regions;

- network-level participation and churn metrics.

The storage layer is designed so that:

- raw data and derived metrics are clearly separated;
- derived views can be recomputed from raw data when models or definitions change;
- historical behaviour can be analysed over different time horizons without re-ingesting external sources.

This separation makes it possible to evolve the analytics and risk models without losing the ability to reproduce previous outputs.

Analytics services

On top of storage, FortisX runs a set of analytics services that compute metrics and indicators needed for the policy engine and for any internal or client-facing tools built on top of the Analytics API. These services:

- transform raw events and baselines into higher-level metrics (for example, validator reliability scores, participation ratios, or network concentration indicators);
- align metrics across networks into a common vocabulary, while preserving network-specific details where they matter;
- maintain current and historical views that are optimised for different use cases, such as:

- comparing validators within a network;

- comparing networks on structural characteristics;

- tracking changes in risk indicators over time.

Analytics services are typically stateless with respect to long-term data; they read from the storage layer, compute metrics or projections, and write updated aggregates back. This keeps their behaviour transparent and makes it easier to audit how particular results were obtained.

Risk modeling layer

The risk modeling layer takes outputs from analytics and organises them into a set of dimensions relevant for staking decisions. Conceptually, it operates as follows:

- **Factor calculation** – metrics are grouped into factors such as technical reliability, concentration and decentralisation, operational behaviour, and infrastructure dependence.
- **Normalisation** – values are normalised into comparable ranges so that they can be combined even when raw units differ between networks or metrics.
- **Scoring and bucketing** – factor scores are combined into overall indicators or buckets (for example, different levels of operational or concentration risk for a validator, pool, or network).

The models and thresholds used here are explicitly configured and versioned. When assumptions change, new versions of the model can be introduced and their impact studied, while preserving the ability to re-run previous versions on the same historical data.

The outputs of this layer are not treated as definitive ratings, but as structured signals that can be used by policies.

Policy engine

The policy engine is the component that translates analytics and risk signals into concrete allocation and rebalancing evaluations and proposals. It operates on three main inputs:

- current allocations across validators, pools, and networks;
- metrics and risk indicators computed by the analytics and modeling layers;
- explicit policy definitions configured by users or governance processes.

Policies are expressed as constraints and rules, such as:

- maximum allowable exposure to a single provider or validator;
- required characteristics for eligible networks (for example, minimum decentralisation indicators);
- exclusion of entities with specific risk profiles;
- limits on how much allocations may change in a given time window.

Given these inputs, the engine produces:

- **allocation evaluations** – assessments of whether current allocations comply with configured policies, and where they diverge;
- **allocation proposals** – suggested changes that would bring allocations back within policy, including specific targets and transition paths (for example, moving part of a position from one set of validators to another).

The policy engine is designed to be deterministic: given the same allocations, data, and policies, it should produce the same output. This property is important for auditability and for reproducing decisions ex post.

External interfaces and execution flow

The results produced by FortisX must be consumable by external systems that hold assets, operate validators, or maintain internal records. To support this, the architecture exposes:

- **dashboards and views** for human operators, showing validator, network, and policy status;
- **alert streams** highlighting events such as large stake movements, changes in risk indicators, or policy breaches;
- **an Analytics API** that allows external systems to query networks, validators, metrics, risk indicators, and policy evaluations;
- **integration hooks** (for example, webhooks or message-based interfaces) through which allocation proposals can be delivered to downstream execution systems.

In a typical flow:

1. FortisX ingests and processes data, updates analytics, and evaluates policies.

2. If policies indicate that allocations should change, the engine produces a set of proposals.

3. Proposals are surfaced via dashboards and interfaces, and received by systems responsible for custody, validator operations, or portfolio management.

4. Those systems apply their own checks and approval processes and, if actions are authorised, execute the necessary on-chain operations.

5. As allocations change, FortisX observes the resulting state and incorporates it into subsequent evaluations.

This separation keeps FortisX focused on analytics and policy logic, while allowing different organisations to implement execution and governance in ways that fit their own requirements.

Resilience and observability

Although detailed operational practices are covered in a later section, certain resilience and observability aspects are part of the architecture itself:

- **Decoupling of components** – ingest, storage, analytics, risk modeling, policy evaluation, and interfaces are separate services connected through well-defined contracts and queues. This reduces the impact of localised failures.
- **Replay and re-computation** – maintaining raw event logs allows models, metrics, and policies to be recomputed if assumptions change or if inconsistencies are detected.
- **Instrumentation** – each component exposes health indicators and performance metrics, making it possible to monitor data freshness, processing latency, and error rates.

These properties are necessary for a platform that is expected to operate continuously across multiple networks, while remaining transparent enough for external teams to review and rely on its outputs. Subsequent sections describe the data model, metrics, models, and operational practices that build on this architectural foundation.

Data model & core entities

The FortisX data model provides a common language for describing validator-based networks, the actors that participate in them, and the signals that matter for staking decisions. It is designed to be explicit, stable, and auditable: every metric, risk indicator, and policy decision is expressed in terms of well-defined entities and relationships.

This section introduces the core objects in the model and how they relate to each other. It does not prescribe a particular physical database schema, but it does fix the conceptual structure that the rest of the platform relies on.

Design goals

The data model is guided by a few principles:

- **Network-agnostic core** – Ethereum, Solana, Polkadot, Avalanche, and Cosmos differ in protocol details, but the same model should apply across them wherever possible.
- **Separation of concerns** – static identifiers and metadata are kept separate from time-varying measurements and derived indicators.
- **Reproducibility** – it must be possible to reconstruct a historical view of networks, validators, and allocations at any given time, based on recorded data and model versions.
- **Policy alignment** – entities and relationships must be rich enough to express realistic allocation and risk policies without depending on ad hoc fields or implicit assumptions.

The entities described below form the backbone of this model.

Networks

A **Network** represents a validator-based protocol (for example, Ethereum, Solana, Polkadot, Avalanche, Cosmos). Each network has:

- a stable identifier used across the platform;
- basic static metadata (name, human-readable code, protocol family, consensus mechanism);
- protocol parameters that influence staking and validation (for example, expected participation behaviour, penalty regimes, key protocol upgrade milestones).

Time-varying network characteristics such as staking participation, concentration, and changes in governance parameters are represented as **metrics** attached to the Network, not as fields on the Network object itself.

Networks act as the top-level scope for other entities: validators, pools, providers, metrics, and policies are all defined within one or more networks.

Validators

A **Validator** is an entity that participates in consensus or block production within a given network. The model distinguishes between:

- a stable **Validator identity** (how FortisX refers to the validator over time); and
- time-varying attributes such as status, performance, and configuration.

Each Validator includes:

- network identifier (which Network it belongs to);
- one or more protocol-specific identifiers (for example, indices, addresses, keys);
- associations to **Providers** or **Pools**, where applicable;
- static metadata where available (self-declared name, description, metadata references).

Metrics such as participation, uptime, penalties, rewards, and configuration changes are stored as time series associated with the Validator. These metrics are the basis for validator-level analytics and risk assessments.

Pools

In networks where stake is aggregated through pools (for example, staking pools, liquid staking protocols, or managed validator sets), FortisX models a **Pool** as a distinct entity:

- a Pool groups one or more Validators under a shared policy, brand, or operational unit;
- delegators may enter or exit positions at the Pool level;
- pool-level metrics (total stake, inflows, outflows, concentration across underlying validators) are derived from validator- and network-level data.

The relationship between Pools and Validators is many-to-many in general: a pool may operate multiple validators, and a validator may, in some networks, be associated with more than one pool or deferred delegation mechanism. The data model makes these relationships explicit rather than assuming a single mapping.

Pools are important for allocation and risk policies that operate at a higher level than individual validators.

Providers

A **Provider** represents an operational entity that runs validators or offers staking as a service. In some cases, a provider corresponds to a well-known operator; in others, it may be an internal construct used to group validators and pools that share infrastructure or control.

A Provider may be associated with:

- multiple Validators across one or more Networks;
- one or more Pools that it operates or supports.

By modelling Providers explicitly, FortisX can:

- track concentration of stake and risk across providers, not only across validators;
- express policies in terms of provider exposure (for example, limits on how much capital may be allocated to a single provider across networks);
- incorporate provider-related events (such as incidents, audits, or configuration changes) into risk assessments.

Provider metadata is intentionally minimal and focused on operational aspects; marketing labels or self-descriptions do not play a role in the data model.

Delegator segments

FortisX does not attempt to model every individual delegator. Instead, it uses **Delegator segments** to represent patterns of stake behaviour where the origin or specific identity is not required for policy decisions.

A DelegatorSegment is defined by:

- the Network and, optionally, Pool or Provider it relates to;
- properties that distinguish behaviour (for example, large vs. small stakes, short-term vs. long-term patterns, specific entry/exit profiles).

Metrics associated with Delegator segments capture:

- flows of stake in and out of Pools, Validators, or Networks;
- reactions to protocol events or changes in performance;
- the emergence of large, concentrated positions that may affect risk.

These segments are used primarily in analytics and risk modeling to understand how the distribution of stake evolves over time and where concentration or liquidity risks may arise.

Events and alerts

Raw behaviour in networks and infrastructure is captured initially as **Events**. Examples include:

- validator participation or missed duties;
- penalties, slashing, or other protocol-enforced actions;

- stake operations (bond, unbond, delegate, redelegate), where visible;
- configuration changes for validators or Pools;
- significant changes in network parameters.

Based on these Events, FortisX generates higher-level **Alerts**. Alerts are structured summaries of conditions that may be relevant for operators or policies, for example:

- large stake inflows or outflows to a Pool or Validator over a short period;
- unusual changes in performance or participation;
- sudden shifts in concentration levels.

Alerts are associated with the relevant entities (Network, Validator, Pool, Provider) and carry enough context to be traceable back to underlying Events and metrics. They are used both in human-facing interfaces and, where appropriate, as triggers in policy evaluations.

Metrics and indicators

Metrics represent quantitative observations over time. In the data model, they are not fields on entities but separate objects that link:

- a **subject** (Network, Validator, Pool, Provider, or DelegatorSegment);
- a **metric type** (for example, participation rate, effective balance, stake share, concentration index);
- a **timestamp or interval**;
- a measured **value**, and optional contextual attributes (such as measurement method or data source).

Derived **indicators** are also modeled explicitly. These are values that combine multiple metrics into a single number or categorical label, for example:

- validator reliability scores based on missed duties and downtime;
- network-level concentration metrics based on stake distribution;
- operational health indicators derived from incident and configuration history.

By treating metrics and indicators as first-class objects, FortisX enables:

- recomputation of indicators when definitions change;
- comparison of the same metric across different subjects and networks;
- explicit links between policies and the metrics they depend on.

Risk profiles

A **Risk profile** is a structured view of how FortisX evaluates a particular entity (Validator, Pool, Provider, or Network) along several dimensions. In the data model, a Risk profile includes:

- the subject it applies to;
- a reference to the **model version** used to compute it;
- values for individual factors (for example, technical reliability, concentration exposure, operational history);
- an overall classification or bucket, where applicable.

Risk profiles are time-stamped and can be stored as a series, allowing FortisX to show how risk assessments evolve as new data arrives or as models are updated.

Policies and external systems can refer to risk profiles explicitly (for example, “exclude validators in certain risk buckets”), and because model versions are recorded, it remains possible to understand which assumptions were in force at the time.

Policies and allocation proposals

Policies and allocation decisions are central to the role of FortisX as a staking platform.

Policies

A **Policy** encodes a set of rules and constraints that define how capital may be distributed. In the data model, a Policy includes:

- scope (which Networks, Providers, Pools, or Validators it applies to);
- references to relevant metrics and risk factors (for example, concentration indicators, reliability scores);
- quantitative limits and thresholds (for example, maximum allocation to a provider, minimum required characteristics for a network);
- metadata such as owner, approval status, and version.

Policies are treated as configuration objects that can be audited and versioned. Changes to a policy are recorded so that historical allocation decisions can be interpreted in context.

Allocation proposals

An **Allocation proposal** represents the outcome of applying Policies to current data. For a given portfolio or capital pool, it includes:

- current allocations across Networks, Providers, Pools, and Validators;
- recommended target allocations that satisfy configured policies;

- a description of the changes needed to reach those targets (for example, amounts to increase or decrease in specific positions);
- the data snapshot and model versions used to derive the proposal.

Allocation proposals are not actions by themselves. They form a bridge between the analytics and policy layers of FortisX and the external systems that execute staking operations. Each proposal can be accepted, rejected, or modified by those systems, and its history can be traced back to the underlying metrics and policies.

Relationships and identifiers

To make the model usable across networks and over time, FortisX maintains a set of internal identifiers that are:

- **stable** – they do not change when external representations (such as addresses or indices) change as a result of protocol upgrades or operational changes;
- **scoped** – identifiers are namespaced by entity type and Network where necessary;
- **traceable** – mappings between internal identifiers and protocol-level identifiers are recorded with validity intervals.

Relationships between entities (for example, Validator–Network, Validator–Provider, Pool–Provider) are explicit and time-aware. This allows the platform to:

- reconstruct historical point-in-time views (which validators belonged to which provider at a given time);
- reason about concentration across different dimensions;
- apply policies consistently even as underlying infrastructure changes.

Summary

The FortisX data model defines a small, focused set of entities—Networks, Validators, Pools, Providers, Delegator segments, Events, Alerts, Metrics, Risk profiles, Policies, and Allocation proposals—and the relationships between them. This structure allows:

- data from heterogeneous networks and providers to be expressed in a common form;
- analytics and risk models to be built on top of a clear and auditable foundation;
- allocation and rebalancing decisions to be expressed as explicit, reproducible objects.

Subsequent sections describe how these entities are populated with metrics and how risk modeling, policies, and operational practices build on top of this model.

Validator metrics

Validator-level metrics are at the centre of FortisX. They describe how individual validators and pools behave over time and provide the raw material for reliability assessments, risk profiles, and allocation policies. This section outlines which aspects of validator behaviour FortisX measures, how those measurements are obtained, and how they are used within the platform.

The goal is not to exhaustively document every network-specific detail, but to define the main categories of metrics that are applied consistently across networks such as Ethereum, Solana, Polkadot, Avalanche, and Cosmos.

Objectives of validator analytics

Validator metrics in FortisX are designed to answer a few practical questions:

- How reliably does a validator participate in consensus and perform expected duties?
- How does its behaviour compare to peers in the same network?
- How has its configuration and operational profile changed over time?
- Which validators or pools exhibit characteristics that may be relevant for concentration, operational, or governance risk?

The metrics described below are grouped into families that support these questions. They are stored as time series and aggregates, enabling both point-in-time evaluation and historical analysis.

Participation and performance

Participation and performance metrics capture how a validator behaves in its primary role within a network.

Typical metrics include:

- **Participation rate** – the fraction of expected duties (for example, attestations, proposals, votes) that the validator has successfully performed over a given interval.
- **Missed duties** – counts or rates of missed attestations, blocks, proposals, or other network-specific responsibilities.
- **Relative performance** – comparisons of a validator's participation or effectiveness to network averages or relevant peer groups.
- **Latency and inclusion patterns**, where observable – for example, how quickly a validator contributes to consensus relative to typical network timings.

These metrics are derived from on-chain events, consensus-layer data, and, where applicable, validator-specific telemetry. They provide a basic view of whether a validator is consistently online and behaving as expected for its role.

Availability and downtime

Availability metrics highlight periods where a validator is not participating as expected. Depending on the network, this may be expressed as:

- **Uptime percentage** over a rolling window or fixed period;
- **Count and duration of downtime episodes**, where the validator's effective participation drops below a configurable threshold;
- **Recovery patterns** – how quickly the validator returns to normal behaviour after incidents or scheduled maintenance.

FortisX does not rely on single-point measurements. Instead, it tracks availability over multiple windows and resolutions, so that both short-term incidents and longer-term trends can be distinguished.

Penalties, slashing, and incidents

Penalties and incidents directly affect the risk profile of a validator or pool. FortisX maintains structured records and metrics for:

- **Protocol-level penalties and slashing events** – including type, severity, and affected balances where this is visible from the network;
- **Patterns of minor penalties** – for example, frequent small penalties that may signal recurring issues even if no major slashing has occurred;
- **Recorded incidents and operator-reported events**, where available – such as operational failures, misconfigurations, or external disruptions.

These signals are combined into time-series metrics and incident logs that feed into the risk model. The aim is not only to mark validators that have suffered major penalties, but also to detect recurring patterns that may indicate operational fragility.

Configuration and software profile

Changes in configuration and software can have important implications for reliability and interoperability. FortisX tracks:

- **Client and version information**, where it can be inferred or reported in a reliable way;
- **Consensus and execution client combinations** in multi-client networks;
- **Changes in configuration** that may affect behaviour or risk (for example, changes in fee or commission parameters, or in operational flags);

- **Rollout patterns for updates**, such as whether a validator upgrades promptly or lags behind the broader network.

Metrics in this family are less about short-term performance and more about assessing whether a validator operates with a discipline that aligns with the expectations of the network and of risk-sensitive delegators.

Economic and fee parameters

FortisX does not model asset prices or future returns, but it does track certain parameters that influence the economic positioning of validators and pools. Typical metrics include:

- **Fee or commission levels** charged by validators or pools, as defined by network-specific mechanisms;
- **Changes in fee parameters over time**, including direction and frequency;
- **Relative positioning** of a validator or pool's fee levels compared to its peers in the same network.

These measurements are used to understand how validators and pools are positioned from an economic perspective, and how frequently they change terms, without attempting to predict or promote specific yield outcomes.

Stake distribution and concentration

From the perspective of both risk and policies, it matters not only how a validator behaves, but also how much stake it attracts.

Validator-level metrics in this area include:

- **Effective stake controlled by the validator** within its network;
- **Share of stake held by the validator within relevant scopes** (for example, within a pool, provider, or the overall network);
- **Inflow and outflow patterns**, where delegation operations are visible, indicating how stake concentration evolves over time.

These metrics inform concentration and decentralisation analyses and are used by policies that limit exposure to individual validators, pools, or providers.

Derived reliability indicators

Beyond raw metrics, FortisX computes derived indicators that summarise aspects of validator behaviour. Examples include:

- **Reliability scores** based on combinations of participation, availability, and incident patterns;
- **Stability indicators** reflecting how often and how strongly performance fluctuates;
- **Configuration stability indicators** capturing the frequency and type of changes to key parameters.

These indicators are not treated as opaque ratings. Their definitions and inputs are documented, and they are stored with references to the underlying metrics and model versions that produced them.

Time scales and granularity

Validator metrics are collected and stored at multiple time scales:

- **Fine-grained measurements** at the level of blocks, slots, or epochs, depending on the network;
- **Aggregated views** over fixed intervals (for example, hourly, daily, weekly), used in dashboards and risk modeling;
- **Rolling windows** that smooth short-term noise while preserving sensitivity to meaningful shifts.

This multi-scale approach allows FortisX to:

- detect short-lived incidents without overreacting to isolated events;
- characterise long-term behaviour and trends;
- recalibrate indicators when network dynamics or model assumptions change.

Data sources and validation

Validator metrics are derived from a combination of:

- consensus-layer and execution-layer data obtained from nodes or indexers;
- publicly accessible RPC endpoints and specialised data services;
- provider- or operator-supplied information where it can be verified or cross-checked.

The platform applies consistency checks where multiple sources exist, and records which sources were used for each metric. When discrepancies are detected, FortisX may mark certain measurements as degraded or exclude them from derived indicators until the underlying issue is resolved.

Use within FortisX

Validator metrics serve several roles inside the platform:

- **Analytics and visualisation** – describing how individual validators and pools behave, how they compare to peers, and how their behaviour changes over time.
- **Risk modeling inputs** – feeding into factor scores and risk profiles that represent dimensions such as technical reliability, operational stability, and concentration exposure.
- **Policy and allocation inputs** – providing the quantitative basis for policies that bound exposure to specific validators or classes of validators and for allocation proposals that reflect those policies.

Because metrics, indicators, and their relationships to policies are recorded explicitly, it is possible to reconstruct why a particular validator was included, excluded, or limited in an allocation proposal at a given point in time.

Subsequent sections extend this view beyond individual validators and pools to network-level and decentralisation metrics, and then to the risk and policy layers built on top of this data.

Network & decentralization metrics

Validator-level metrics describe how individual actors behave. Network-level and decentralization metrics describe the environment in which they operate: how stake is distributed, how concentrated control has become, and how the overall system evolves over time. FortisX combines both perspectives when assessing risk and designing allocation policies.

This section outlines the main categories of network-level metrics that FortisX maintains across validator-based networks such as Ethereum, Solana, Polkadot, Avalanche, and Cosmos, and how these metrics are used within the platform.

Objectives of network-level analytics

Network and decentralization metrics in FortisX are designed to answer questions such as:

- How is stake distributed across validators, pools, and providers?
- How concentrated is effective control in practice, and how is this changing over time?
- How active is the network in terms of participation, governance, and protocol-level events?
- Are there emerging patterns that may affect the risk of running or delegating stake within a given network?

These questions are addressed through metrics that capture participation, stake distribution, concentration, churn, and indicators related to governance and protocol changes.

Participation and staking participation

At the network level, FortisX tracks overall staking and participation characteristics, including:

- **Total staked amount** – the aggregate quantity of stake that is actively participating in validation according to network rules.
- **Staking participation ratio** – the ratio of staked stake to relevant supply measures (for example, a subset of circulating or eligible supply, depending on network definitions).
- **Effective participation** – how much of the staked set is actively contributing to consensus versus being nominally staked but frequently unavailable.

These metrics help characterise how much of a network's potential validation capacity is actually used, and how sensitive the network might be to concentrated failures or participation drops.

Stake distribution across validators, pools, and providers

The distribution of stake is a central input for decentralization and concentration analysis. FortisX maintains time-series metrics on:

- **Stake share by validator** – each validator's percentage of the network's total active stake.
- **Stake share by pool** – in networks or setups where pools are present, the share of stake aggregated at each pool.
- **Stake share by provider** – aggregated stake across validators and pools operated by the same provider, across one or multiple networks where relevant.

These metrics are computed at regular intervals and stored historically, so that FortisX can observe how distribution patterns change, whether some entities gain disproportionate share, and how quickly those changes occur.

Concentration and decentralization indicators

Using stake distribution metrics, FortisX derives indicators that describe how concentrated or diffuse control in a network appears to be. Examples of such indicators include:

- **Share of stake in top N validators, pools, or providers** – for several values of N, providing a simple view of how much stake is held by the largest actors.
- **Concentration indices** – such as measures analogous to the Herfindahl–Hirschman Index (HHI) or other dispersion metrics computed over stake shares.
- **Tail distribution characteristics** – such as how quickly stake share declines beyond the top tier of validators or pools.

These indicators do not attempt to define "good" or "bad" levels of decentralization in isolation. Instead, they provide a quantitative basis for comparing networks, monitoring changes within a network, and expressing policies that depend on concentration levels.

Churn and stake mobility

Static snapshots of stake distribution only show part of the picture. FortisX also tracks **churn** and stake mobility:

- **Validator and pool entry/exit rates** – how often validators or pools enter or leave the active set.
- **Stake inflows and outflows** at the network, pool, and provider levels over defined time windows.
- **Persistence of stake allocations** – how stable stake positions have been, and whether there are frequent reallocations between a small number of destinations.

Churn metrics help identify environments where stake is highly mobile or where a few destinations periodically absorb disproportionate inflows. This has implications for both operational risk and for

the stability of allocation policies.

Activity, load, and protocol events

In addition to stake distribution and concentration, FortisX observes indicators related to overall network activity and load, for example:

- **Transaction and block activity metrics**, where relevant to the performance and reward environment for validators.
- **Governance and protocol events**, such as parameter changes, upgrades, or votes that may affect staking conditions or validator operations.
- **Network-level incident signals**, such as extended finality delays, elevated reorganisation rates (if applicable), or coordinated responses to known issues.

These indicators are used to provide context for validator and stake behaviour. For example, a sudden change in stake concentration may follow a protocol event, or an increase in minor penalties may correlate with a period of elevated network stress.

Cross-network comparability

Different networks use different units, epochs, and mechanisms. FortisX does not attempt to flatten these differences into a single synthetic score, but it does align metrics sufficiently to support structured comparisons:

- core concepts such as **stake share**, **concentration indices**, and **churn rates** are defined in a network-agnostic way;
- network-specific details (such as epoch length, penalty regimes, or governance mechanisms) are recorded and used to interpret metrics correctly;
- derived indicators can be grouped into common dimensions (for example, concentration, churn, activity), while preserving per-network nuances in documentation and metadata.

This approach enables policies that impose similar structural constraints across networks (for example, limits on acceptable concentration levels) without assuming that all networks behave identically.

Data sources and quality considerations

Network-level and decentralization metrics are derived from:

- on-chain state and events obtained from full nodes or indexers;
- observed validator sets and active stake distributions at relevant intervals;

- protocol metadata and governance data where accessible and reliable.

As with validator metrics, FortisX records the provenance of network metrics and applies consistency checks where multiple sources are available. When upstream data is incomplete or ambiguous, the platform may:

- mark certain metrics as degraded or provisional;
- delay the use of specific indicators in risk or policy evaluations until data quality improves;
- record data quality status alongside metrics for downstream consumers.

This ensures that allocations and risk assessments can take into account both the measured properties of a network and the reliability of the measurements themselves.

Use within FortisX

Network and decentralization metrics flow into several parts of the platform:

- **Analytics and monitoring** – providing views of how each supported network evolves in terms of stake distribution, concentration, churn, and activity.
- **Risk modeling** – contributing to factors that reflect concentration risk, governance and protocol stability, and environment-level conditions that affect validators and pools.
- **Policy design and evaluation** – enabling policies that:

- restrict exposure to networks with certain concentration characteristics;

- require minimum levels of decentralization or participation for eligibility;

- adapt allocation decisions in response to sustained changes in network structure.

Because network-level metrics are recorded as time series and tied to specific model and policy versions, FortisX can reconstruct which network conditions and assumptions were in place when particular allocation decisions were made. This is important for both internal governance and external review.

The next sections describe how these metrics combine with validator-level data in the risk modeling layer, and how the resulting signals are used by the policy engine to shape allocations and rebalancing behaviour.

Risk modeling

Risk modeling in FortisX provides a structured way to interpret validator and network metrics. Instead of treating each metric in isolation, the platform organises them into a set of factors that are relevant for staking decisions and then aggregates those factors into risk profiles for validators, pools, providers, and networks.

This section describes the objectives of the risk model, how it is built from underlying data, how it is versioned and interpreted, and how it interacts with policies and allocations.

Objectives

The risk model in FortisX is designed to answer questions such as:

- Which validators, pools, or providers exhibit patterns that warrant caution, closer monitoring, or explicit limits?
- How do network-level conditions (such as concentration or churn) influence the environment in which validators operate?
- How can quantitative signals be organised so that allocation policies are based on observable structure rather than informal judgement?

The model is not intended to predict prices or guarantee outcomes. Its purpose is to make relevant dimensions of risk explicit, measurable, and comparable over time.

Inputs to the model

The risk model consumes a broad set of inputs from other layers of the platform:

- **Validator metrics** – participation, availability, penalties and incidents, configuration changes, fee parameters, stake share, and stability over time.
- **Network and decentralization metrics** – stake distribution, concentration indicators, churn, governance and protocol events, and measures of network activity and load.
- **Provider-level information** – aggregations of validator and pool metrics across a provider's footprint, including cross-network concentration where applicable.
- **Events and alerts** – structured summaries of notable conditions (for example, large stake movements, recurring minor penalties, or sudden changes in participation).

These inputs are aligned in time and tagged with model and data provenance, so that risk profiles can be reconstructed under different assumptions if needed.

Risk factors

Rather than collapsing all information into a single score, FortisX organises risk into a set of **factors**, each representing one dimension of interest. Typical factors include:

- **Technical reliability**

Based on participation, availability, missed duties, penalty patterns, and the frequency and impact of incidents.

- **Operational stability**

Reflecting configuration and software changes, rollout discipline for updates, and the presence of recurring operational issues.

- **Concentration and exposure**

Capturing stake share, distribution across validators, pools, and providers, and the share of total stake controlled by a given entity or group.

- **Network environment**

Summarising aspects of the underlying network that affect all participants, such as churn, structural concentration levels, and the incidence of protocol-level events.

- **Infrastructure dependence**

Where observable, indicating how reliant a validator, pool, or provider is on particular pieces of infrastructure or service providers.

Depending on the subject (Validator, Pool, Provider, Network), not all factors are equally relevant or available; the model accounts for this when constructing risk profiles.

Normalisation and scaling

Underlying metrics often differ in units and ranges across networks. To make them combinable within a factor, FortisX applies a normalisation layer that:

- converts raw values into dimensionless quantities on a common scale (for example, between 0 and 1 or within a small integer range);
- incorporates network-specific context where necessary (for example, typical performance bounds in a given network);
- records the transformation used, so that the mapping from raw metrics to normalised values remains transparent.

Normalisation parameters and methods are part of the model definition and are versioned alongside it.

Factor scores and aggregation

For each subject and factor, the model produces a **factor score**:

- factor scores are computed from one or more normalised metrics using simple, documented combinations (for example, weighted sums, thresholds, or bounded functions);
- where data is incomplete or degraded, the factor score may be marked as partial or unavailable rather than forced into a default value;
- intermediate values used to compute factor scores are recorded so that the contribution of each metric can be inspected.

Factor scores may then be aggregated into higher-level views, depending on use case:

- **Risk profiles** – structured sets of factor scores associated with a subject and a specific model version;
- **Buckets or categories** – coarse labels (for example, low / moderate / elevated risk along a factor) derived from ranges of factor scores for use in policies.

Aggregation rules and bucket boundaries are configured and versioned, not embedded implicitly in code.

Model versions and evolution

Networks, staking practices, and available data sources evolve over time. The FortisX risk model is therefore explicitly **versioned**:

- each version defines:

- which metrics and indicators are used;

- how they are normalised;

- how they are combined into factor scores and buckets;

- risk profiles are tagged with the model version that produced them;
- changes to the model (for example, altered weights, additional metrics, or new factors) yield new model versions rather than silently modifying the behaviour of existing ones.

This approach allows FortisX to:

- evaluate the impact of prospective model changes on historical data before adopting them;
- reconstruct the view of risk that existed at a given time under a specific model version;
- provide clear context when explaining why an allocation decision was made.

Interpretation and limitations

Risk profiles and factor scores are designed to support structured reasoning, not to replace it. In particular:

- a higher or lower score along a factor should be interpreted with reference to its definition, inputs, and model version;
- missing or degraded data is treated explicitly, and such cases may warrant further investigation rather than automatic inclusion or exclusion;
- different organisations may assign different weights or thresholds to the same factors in their internal policies.

The model does not assert that one validator, pool, provider, or network is “safe” or “unsafe” in an absolute sense. It provides a means to compare and track conditions under a given set of assumptions and to express policies that are consistent with those assumptions.

Use in policies and allocations

The main consumer of risk modeling output inside FortisX is the **policy engine**. Policies may refer to:

- specific factor scores (for example, requiring a minimum level of technical reliability);
- combinations of factors (for example, allowing exposure only where both operational stability and network environment factors are within defined ranges);
- buckets or categories (for example, excluding subjects in designated high-risk categories, or limiting their aggregate share).

When the policy engine evaluates current allocations or constructs allocation proposals, it uses risk profiles to:

- identify entities that are in or out of policy based on their risk characteristics;
- compute maximum permissible exposure to particular validators, pools, providers, or networks;
- determine where reallocations could reduce concentration or align allocations more closely with defined risk tolerances.

Because risk profiles, policies, and data snapshots are all versioned and time-stamped, it is possible to explain how specific allocation proposals arose and how risk considerations were taken into account.

Summary

The FortisX risk model organises a diverse set of validator and network metrics into a coherent structure of factors, scores, and profiles. By making each step of the transformation explicit and versioned, it provides:

- a repeatable way to interpret complex data about validators, pools, providers, and networks;
- a stable interface between analytics and policy logic;

- a basis for auditability and explanation of allocation and rebalancing decisions.

Subsequent sections describe how the policy engine uses these risk signals together with explicit rules and constraints to shape staking allocations and rebalancing behaviour across networks.

Policy engine & allocation rules

The policy engine is the component of FortisX that turns observable data and risk assessments into concrete decisions about how capital may be allocated and how positions should be rebalanced. It does not move assets itself; instead, it evaluates allocations against explicit rules and produces proposals that can be acted on by external systems responsible for execution.

This section describes what a policy is in FortisX, how policies are evaluated, which inputs they use, how allocation proposals are constructed, and how the overall process remains deterministic and auditable.

Objectives

The policy engine is designed to:

- express staking and allocation rules in a **formal, machine-readable** way;
- ensure that current allocations are **measured** against those rules, using the same metrics and risk model defined elsewhere in the platform;
- provide **concrete proposals** for reallocating capital when allocations drift away from the defined policies;
- make every step of this process **traceable and reproducible**, so that decisions can be reviewed by operators and governance bodies.

The engine acts as the link between data and action: it consumes analytics and risk signals, and produces structured recommendations rather than informal guidance.

Inputs to the policy engine

The policy engine operates on three main classes of input:

1. **Current allocations**

A snapshot of how capital is presently distributed across:

- Networks;

- Providers;

- Pools;

- Validators.

Allocations may be represented at different levels of granularity depending on the use case (for example, per portfolio, per client, or per internal book).

2. **Analytics and risk profiles**

The metrics and risk outputs described in earlier sections:

- validator-level metrics and derived indicators;

- network and decentralization metrics;

- provider-level aggregations;

- risk profiles for each subject, including factor scores and buckets.

3. **Policies**

A set of configuration objects that specify:

- what exposures are allowed or disallowed;

- which entities are eligible or ineligible;

- how quickly allocations may change;

- how policy breaches should be flagged.

Policies are versioned and time-stamped, just like data and models, so that evaluations can be understood in context.

Policy structure

In FortisX, a Policy is a structured object rather than a free-form script. While the exact configuration language may evolve, policies are conceptually composed of:

- **Scope**

The set of subjects the policy applies to, such as:

- one or more Networks;

- a class of Providers or Pools;

- specific portfolios or allocation books.

- **Eligibility rules**

Conditions that define when an entity may be considered for allocation, for example:

- minimum technical reliability factors;

- acceptable ranges for concentration or decentralization indicators;

- exclusion of entities in certain risk buckets or with specific incident histories.

- **Exposure limits**

Quantitative constraints on how much capital can be allocated:

- maximum share of a portfolio in a single Validator, Pool, Provider, or Network;

- bounds on aggregate exposure to a group of entities sharing characteristics (for example, validators operated by the same provider, or networks in a given protocol family).

- **Rebalancing rules**

Parameters that govern how quickly and how far allocations may change:

- maximum change in allocation to a subject over a time window;

- thresholds for when a deviation from target allocations triggers a proposal;

- rules for gradual transitions versus immediate corrections.

- **Alerting and escalation hooks**

Definitions of when policy breaches or high-severity changes in risk should produce alerts, require manual review, or trigger specific workflow steps in external systems.

By structuring policies in this way, FortisX ensures that they can be interpreted and evaluated consistently, and that their effect on allocations can be analysed in advance.

Evaluation process

Policy evaluation in FortisX proceeds in two main phases: **compliance checks** and **proposal generation**.

Compliance checks

In the compliance phase, the engine:

1. Loads the current allocation snapshot for the relevant scope.

2. Fetches the corresponding analytics and risk profiles for all subjects under that scope.

3. Evaluates each applicable Policy against:

- current exposures;

- eligibility criteria;

- any referenced risk profiles or metric thresholds.

The output of this phase is:

- a set of **policy evaluation records** that indicate, for each subject and policy:

- whether the policy is currently satisfied;

- the magnitude and direction of any deviations (for example, by how much an exposure exceeds a limit);

- any associated alerts that should be emitted.

These records can be consumed both by dashboards and by external systems that monitor policy adherence.

Proposal generation

Where policies are not satisfied and rebalancing is permitted, the engine can move to proposal generation:

1. Computes feasible target allocations that:

- bring exposures back within policy-defined bounds;

- respect any constraints on maximum change per interval;

- preserve other invariants, such as total allocated capital within a portfolio.

2. Constructs **allocation proposals** that describe:

- new target weights or amounts for each subject;

- the net increase or decrease required relative to the current state;

- any sequencing or phasing information, if rebalancing is intended to occur in steps.

3. Attaches metadata to each proposal, including:

- which policies were involved;

- which data snapshot and model versions were used;

- evaluation timestamps and identifiers.

Allocation proposals are deterministic given the inputs: running the engine with the same data and policies yields the same result.

Determinism and auditability

Determinism is a key property of the policy engine:

- For a given combination of:

- allocation snapshot;

- analytics and risk profiles;

- policy set and model versions;

- the engine produces **the same evaluations and proposals**.

This supports:

- **Auditability** – it is possible to reconstruct the reasoning behind a past allocation proposal by re-running the engine with historical data and configurations.
- **Explainability** – operators and risk committees can inspect which metrics, risk factors, and policy clauses were active and how they contributed to a given proposal.
- **Testing and simulation** – model and policy changes can be applied to historical data in a non-production environment to understand their impact before adoption.

To facilitate this, all relevant inputs and outputs are stored with identifiers and timestamps, and references between them are recorded explicitly.

Governance and policy lifecycle

Policies themselves are subject to governance and lifecycle management. In FortisX, policies:

- are created, updated, and retired through controlled processes;
- carry metadata such as:

- author or owner;

- creation and modification timestamps;

- approval status and, where applicable, governance references;

- are versioned, so that changes result in new policy versions rather than silent edits in place.

The policy engine always evaluates allocations against a **specific set of policy versions** that is in force at the evaluation time. When policies are updated, the resulting change in behaviour can be analysed by comparing the outputs of the engine under old and new versions, using the same historical data.

Interaction with external execution systems

FortisX does not assume custody of assets or direct control over validator keys. Instead, external systems are responsible for:

- approving or rejecting allocation proposals;
- translating proposals into concrete execution steps (for example, on-chain staking operations or internal booking changes);
- enforcing any additional internal rules, approvals, or compliance checks.

The policy engine supports this integration by:

- exposing evaluations and proposals through dashboards, APIs, and integration hooks;
- structuring proposals in a way that is compatible with typical execution flows (for example, specifying target positions rather than assuming step-by-step transactions);
- recording the identifiers necessary for external systems to correlate proposals with their own events and decisions.

This separation allows organisations to embed FortisX into their existing governance and operational frameworks while still benefiting from a consistent analytic and policy layer.

Examples of policy types

While detailed examples are outside the scope of this section, the following high-level patterns illustrate how policies can be composed:

- **Concentration limits**

Limit the share of a portfolio that may be allocated to:

- a single validator or pool;

- a single provider across all networks;

- a single network within a broader multi-network strategy.

- **Eligibility filters**

Exclude entities that:

- fall into certain risk buckets along a specific factor (for example, persistent operational issues);

- have unresolved incidents of a particular type;

- do not meet defined decentralization or participation criteria at the network level.

- **Change control**

Restrict how fast allocations can be adjusted:

- maximum relative or absolute change per interval;

- minimum time between major reallocations under normal conditions.

- **Event-driven safeguards**

Require additional review or suspend certain types of changes when:

- severe incidents are detected at a provider or network level;

- specific governance or protocol events occur.

These patterns can be combined and specialised to reflect different risk appetites and operational constraints without changing the core behaviour of the policy engine.

Summary

The FortisX policy engine provides a formal link between data and allocation decisions. It:

- consumes current allocations, analytics, and risk profiles;
- evaluates explicit, versioned policies against this data;
- produces deterministic, auditable allocation evaluations and proposals;
- leaves execution and custody to external systems that operate under their own governance.

By separating analytics, risk modeling, policies, and execution, FortisX allows organisations to automate parts of their staking allocation process while retaining control over how decisions are implemented and reviewed.

Operations & reliability

FortisX is designed to run as a long-lived, continuously operating platform that observes multiple validator-based networks and supports allocation and rebalancing decisions. This requires operational practices that focus not only on service availability, but also on data freshness, integrity, and the ability to explain behaviour over time.

This section describes the operational objectives and reliability model of FortisX: how the platform is monitored, how incidents and data issues are handled, how changes are introduced, and how continuity is maintained in the presence of dependencies such as networks, providers, and infrastructure.

Operational objectives

Operationally, FortisX is built around a small set of objectives:

- **Service availability** — the core services responsible for ingest, analytics, risk modeling, policy evaluation, and external interfaces should remain available and responsive under normal conditions and degrade in a controlled manner under stress.
- **Data freshness** — metrics and risk assessments should reflect network and validator conditions with bounded delay, subject to the capabilities of upstream data sources.
- **Data integrity and reproducibility** — data used for analytics and policies should be internally consistent, and it should be possible to reconstruct past views using recorded inputs and model versions.
- **Transparent degradation** — when external dependencies fail or degrade, the impact on FortisX should be visible, and affected outputs should be marked accordingly rather than silently treated as complete.

These objectives shape the design of monitoring, incident handling, and change management across the platform.

Availability and data freshness

FortisX distinguishes between two related but separate dimensions:

- **Service availability** – whether core components (ingest, storage, analytics, risk modeling, policy engine, APIs) are reachable and performing within expected bounds.
- **Data freshness** – how recent the underlying metrics and risk profiles are relative to current network conditions.

The platform tracks both:

- For each supported network, FortisX maintains indicators of:

- the most recent block, slot, or epoch observed;

- the lag between network head and ingested data;

- the recency of derived metrics and risk profiles.

- For each core service, FortisX tracks:

- request success rates and latencies;

- internal queue depths and processing latencies;

- resource utilisation and error rates.

Users and external systems can inspect both kinds of information, so they can distinguish between an unavailable service and a service that is available but operating with stale or degraded data for a particular network.

Monitoring and alerting

Monitoring is integrated into each layer of the architecture:

- **Ingest and data pipeline**

- connectivity and error rates for upstream sources (nodes, RPC providers, indexers);

- progress metrics (for example, last processed height) per network and per collector;

- validation and quality checks, such as detection of missing or inconsistent data segments.

- **Storage and aggregation**

- health of underlying storage systems;

- success rates and performance of aggregation jobs;

- capacity indicators and growth rates.

- **Analytics and risk modeling**

- timing and success of metric computation and risk profile updates;

- detection of unexpected discontinuities or anomalies in derived metrics.

- **Policy engine and interfaces**

- frequency and success of policy evaluations;

- volumes and latency of API calls and integration events;

- error conditions in dashboards, APIs, and downstream delivery.

Alerting rules are defined for conditions such as:

- sustained lag in data ingest for a network;
- repeated failures in a collector or aggregation job;
- anomalies in data volumes or distributions that may indicate upstream or internal issues;

- degradation in API or dashboard responsiveness.

Alerts are routed to operational teams for investigation and, where necessary, escalation.

Incident response

When incidents occur, FortisX follows a structured process that separates **technical remediation** from **post-incident analysis**.

Typical steps include:

1. **Detection and triage**

- Confirming whether the alert corresponds to a real issue.

- Classifying the incident (for example, ingest disruption, data quality issue, analytics failure, interface degradation).

2. **Containment and mitigation**

- Taking steps to prevent the issue from propagating (for example, isolating faulty data sources or pausing specific jobs).

- Providing clear signals to downstream systems (for example, marking affected metrics as degraded or temporarily withholding certain outputs).

3. **Resolution**

- Restoring normal behaviour of affected components.

- Initiating backfill or recomputation processes where data has been skipped or corrupted.

4. **Post-incident review**

- Documenting the timeline, root causes, and impact.

- Identifying corrective actions, such as improved validation, additional monitoring, or architectural adjustments.

Where incidents have a visible impact on external consumers, FortisX records and communicates the scope of the incident so that users can interpret affected analytics and decisions in context.

Change management and releases

The platform is designed to evolve over time: new networks may be added, models and metrics may be refined, and policies and interfaces may be extended. To manage this safely, FortisX applies disciplined change management:

- **Environment separation**

Changes are developed and tested in non-production environments that mirror the structure of the production system as closely as practical.

- **Incremental rollout**

New features or model versions may be introduced gradually:

- running in shadow mode alongside existing versions;

- processing real data but not yet driving decisions;

- with explicit comparison of outputs before activation.

- **Configuration and model versioning**

Risk models, policies, and key configuration parameters are versioned, with clear delineation between code changes and configuration changes.

- **Release records**

Releases are accompanied by records that describe:

- what has changed;

- which components and networks are affected;

- any expected impact on metrics, risk profiles, or policy outputs.

When changes affect the semantics of metrics, risk scores, or policies, these changes are reflected in documentation and, where necessary, in the whitepaper itself.

Data backfills and corrections

Because FortisX depends on external data sources and complex processing, there are cases where:

- data is temporarily missing or incomplete;
- upstream providers later correct or revise information;
- model definitions change in ways that require recomputation.

To handle this, the platform supports controlled **backfills** and **recomputations**:

- Raw event logs and snapshots are retained for periods sufficient to reconstruct relevant historical views.
- Backfill jobs can be run for specific time ranges, networks, or entities to close gaps or incorporate corrected upstream data.
- Derived metrics, risk profiles, and policy outputs can be recomputed based on the same raw inputs and model versions, or under updated models where analysis requires it.

Backfills and corrections are tracked so that users can see when historical data or indicators have been updated and under which model or configuration.

Dependency management and external providers

FortisX depends on several categories of external components:

- validator-based networks themselves;
- RPC providers and indexing services;
- infrastructure platforms (for example, compute, storage, networking);
- optional third-party data services.

Operational practices include:

- avoiding single points of dependency where possible (for example, using multiple data providers for critical networks);
- monitoring the health and behaviour of upstream services and reacting when they fail or deviate from expected patterns;
- maintaining clear interfaces so that providers can be changed or reconfigured without redesigning core services.

When external dependencies experience outages or behavioural changes, FortisX aims to:

- constrain the impact to the networks or features directly affected;
- maintain internal consistency by marking certain metrics or outputs as degraded rather than extrapolating from incomplete data.

Operational transparency for users

FortisX is intended to be part of broader operational and risk frameworks. To support this, the platform exposes operational signals to users and integrating systems, such as:

- indicators of data freshness and quality for each network;
- status of core platform services and interfaces;
- summaries of recent incidents that have affected data or service availability;
- version information for risk models and policies currently in force.

This operational transparency allows organisations to:

- incorporate FortisX status into their own monitoring and incident processes;
- align internal governance (for example, risk committees, change advisory boards) with changes and events in the platform;
- interpret allocation proposals and analytics in light of the operational conditions under which they were produced.

Summary

Operational discipline and reliability are central to the role of FortisX as a staking and analytics platform. The system is designed to:

- separate concerns between ingest, analytics, risk modeling, policies, and interfaces;
- monitor each component and its external dependencies;
- handle incidents, data gaps, and model changes in a controlled and traceable way;
- expose sufficient operational information for users to rely on its outputs within their own control frameworks.

The next section describes how these operational practices are complemented by security measures, audits, and compliance considerations that govern how FortisX is built and run.

Security, audits & compliance

In this chapter, “FortisX” refers specifically to the analytics and policy layer described earlier: although it does not by default custody client assets or operate validators on behalf of users, it still handles sensitive data and influences decisions with financial consequences, so its own security posture must treat the platform as critical infrastructure.

This section outlines how FortisX approaches security at the platform level, how independent reviews and audits fit into that approach, and how the system is designed to cooperate with the compliance frameworks of organisations that use it.

Security objectives and threat model

The security model of FortisX is built around a few objectives:

- **Integrity of analytics and policies** — metrics, risk profiles, and policy evaluations must not be altered without detection. Decisions based on FortisX should rely on data and logic that are protected from unauthorised modification.
- **Confidentiality of sensitive information** — any non-public data held by FortisX—such as internal configuration, API keys, integration metadata, or user profiles—must be protected in transit and at rest.
- **Controlled influence over allocations** — because FortisX produces allocation and rebalancing proposals, it must be clear who can influence the models, policies, and configurations that generate those outputs.
- **Separation from custody and execution** — key material and execution rights over client assets remain outside the platform unless explicitly agreed; FortisX is not assumed to be a signing or custody system.

The threat model includes:

- attempts to tamper with analytics, risk models, or policy configurations;
- unauthorised access to operational interfaces or integration endpoints;
- exploitation of software or infrastructure vulnerabilities in the platform;
- misuse of legitimate access (for example, misconfigured privileges or weak governance around policy changes).

Separation of duties

A central design principle in FortisX is **separation of duties** between:

- **Analytics and data acquisition** – ingest and computation of metrics and risk profiles.
- **Policy definition** – configuration of rules and constraints that govern allocations.

- **Execution systems** – external platforms that hold keys, manage validators, or record positions.

This separation is enforced both technically and organisationally:

- Components that ingest and store data are isolated from components that manage policy configuration.
- Policy configuration interfaces are restricted to users and roles with explicit authorisation.
- Execution systems integrate with FortisX through well-defined interfaces and maintain their own approval and control processes.

As a result, a compromise of one area (for example, ingest infrastructure or a specific integration) does not automatically grant the ability to alter policies or move assets.

Access control and authentication

Access to FortisX components and data is controlled through layered mechanisms:

- **Role-based access control (RBAC)**

- System functions are grouped into roles (for example, read-only analytics, policy administration, integration management).

- Users and service accounts are granted only the permissions required for their tasks.

- **Strong authentication for privileged access**

- Administrative and policy-related interfaces require multi-factor authentication where available.

- Access from automation or integration services uses scoped credentials (for example, API keys or service identities) with clearly defined privileges.

- **Segregation of environments**

- Production environments are separated from development and test environments.

- Non-production environments use separate credentials and data, and are not permitted to affect production allocations or configurations.

Changes to access rights are logged and reviewed as part of operational governance.

Data security and privacy

FortisX is not designed to store private keys or client assets. Nevertheless, it handles other categories of data that require protection:

- **Configuration and integration metadata**

- Information about network endpoints, providers, and integrations is treated as sensitive.

- Secrets such as API keys, database credentials, and integration tokens are stored in dedicated secret management systems, not in code or plain configuration files.

- **Analytics and operational data**

- Metrics, logs, and risk profiles are stored in infrastructure that supports encryption at rest and controlled access.

- Communication between components and with external systems is protected with transport-level encryption.

- **User and organisational data**

- Where the platform stores information about users or client organisations, collection is limited to what is operationally necessary.

- Retention and access rules are aligned with those needs and with applicable privacy requirements.

Data handling practices are documented so that client security and compliance teams can review how FortisX fits into their own policies.

Secure development and configuration management

Security considerations are integrated into the software development and configuration lifecycle:

- **Code and configuration review**

- Changes to code and critical configuration (including risk models and policy frameworks) are reviewed by more than one person before deployment.

- Automated checks (such as static analysis and dependency scanning) are applied to code repositories where practical.

- **Dependency management**

- Third-party libraries and tools are monitored for security advisories.

- Updates are applied in a controlled manner, with testing in non-production environments before release.

- **Configuration as data**

- Risk model definitions, policy templates, and other core behaviours are represented as structured data (for example, configuration or model definitions) rather than ad hoc code changes.

- This makes it easier to review, version, and audit changes that affect how allocations and risk assessments are computed.

Infrastructure security

The infrastructure that runs FortisX is secured with standard hardening and isolation practices:

- **Network segmentation**

- Internal services are not directly exposed to the public internet unless required.

- Administrative interfaces are restricted to controlled networks or VPNs.

- **System hardening and patching**

- Operating systems and platform components are kept up to date with security patches.

- Default services and ports that are not needed are disabled.

- **Backups and recovery**

- Critical data stores are backed up regularly.

- Restoration procedures are tested periodically to ensure that data can be recovered in a controlled manner.

- **Logging and audit trails**

- Security-relevant events (such as authentication attempts, policy changes, configuration updates, and administrative actions) are logged.

- Logs are protected from tampering and retained for periods sufficient to support investigations and compliance reviews.

Independent security assessments and audits

Internal controls are complemented by independent reviews from specialised security firms. Selected components of the FortisX stack have been, and are expected to continue to be, reviewed by external providers such as **Certik** and **Cyberscope**, alongside other qualified auditors, with a scope that can include:

- **Application and infrastructure security reviews**

- Third-party security firms review components of the platform, including APIs, administrative interfaces, and deployment architecture.

- **Model and policy auditability**

- Reviews that focus on how risk models and policies are defined, versioned, and enforced, and whether they can be reliably reconstructed over time.

- **Operational and process assessments**

- Evaluations of incident response, change management, and access control processes.

Public artefacts of these reviews are referenced in the legal and documentation bundle, for example:

- **Security Assessment (Certik)** – Audit report
- **Security Assessment (Cyberscope)** – Audit report

The scope, frequency, and findings of such assessments depend on the maturity of the platform and the requirements of its clients. Where appropriate, summaries or attestations (for example, from Certik, Cyberscope, or similar firms) may be shared with users under agreed conditions so they can incorporate them into their own due diligence.

External assessments do not replace internal responsibility for security; they provide an additional layer of scrutiny and feedback.

Compliance alignment

FortisX is designed to be integrated into environments where formal governance and compliance processes are in place, especially for institutional users. While the platform itself is not positioned as a regulatory framework, it supports compliance efforts by:

- providing **clear audit trails** for:

- changes to risk models and policies;

- allocation evaluations and proposals;

- operational incidents and resolutions;

- enabling **segregation of duties** between teams that define policies, operate infrastructure, and approve or execute allocations;
- exposing **versioned documentation and configuration** so that governance bodies can see which assumptions were in effect at a given time.

Regulatory obligations vary between jurisdictions and organisations. FortisX does not attempt to define or satisfy those obligations on behalf of its users; instead, it aims to provide artefacts and controls that can be incorporated into existing frameworks.

Summary

Security in FortisX is approached as a combination of:

- architectural decisions (separation of duties, non-custodial design);
- technical controls (access control, encryption, hardening, monitoring);
- process discipline (review, change management, incident response);
- independent review (external security assessments and audits by firms such as Certik, Cyberscope, and other qualified providers where appropriate).

Together, these measures are intended to ensure that analytics, risk models, and policy evaluations can be relied upon as part of a broader staking and governance framework, while keeping control over assets and execution with the organisations that use the platform.

Analytics API & roadmap

The Analytics API is the primary interface through which external systems access the same data and signals that FortisX uses internally for analytics, risk modeling, and policy evaluation. It is designed to be predictable, versioned, and suitable for integration into monitoring systems, internal dashboards, risk engines, and execution platforms that participate in staking across validator-based networks.

This section describes the role of the Analytics API, the main data domains it exposes, how it fits into typical integration patterns, and how FortisX is expected to evolve over time in terms of coverage, models, and interfaces.

Role of the Analytics API

Internally, FortisX maintains a continuous data pipeline and a set of models that produce metrics and risk profiles for networks, validators, pools, and providers, and that generate allocation evaluations and proposals. The Analytics API exposes a structured view of these outputs so that:

- operators and teams can build their own dashboards and monitoring around FortisX data;
- risk and governance systems can incorporate metrics, risk factors, and policy evaluations into their own workflows;
- execution platforms can receive allocation proposals and context, then apply their own controls before taking action.

The API does not attempt to encapsulate all operational logic. Instead, it presents clear, queryable representations of entities, metrics, risk profiles, and policy outcomes that can be combined with local information and processes.

Design principles

Several principles guide the design of the Analytics API:

- **Consistency with the data model** — endpoints reflect the conceptual model described elsewhere in this document: Networks, Validators, Pools, Providers, Metrics, Events, Alerts, Risk profiles, Policies, and Allocation proposals.
- **Read-focused and side-effect free** — the Analytics API is primarily read-only from the perspective of external systems. Configuration and policy management are handled through controlled channels rather than general-purpose public endpoints.
- **Deterministic responses** — given a defined snapshot and model version, the same query yields the same result. Where data changes over time, responses are tied to timestamps or versions so that consumers can reason about evolution.
- **Versioning and stability** — changes to response structures or semantics are introduced through explicit versioning, rather than silent modifications of existing endpoints.

- **Transparency of provenance** — responses include references to data sources, model versions, and policy versions where relevant, so that downstream systems can evaluate and log not only values but also their context.

Data domains

At a high level, the Analytics API is organised around a small set of domains corresponding to core entities and outputs in FortisX.

Networks

Network-oriented endpoints provide:

- identifiers and metadata for supported validator-based networks;
- high-level participation, stake distribution, and concentration metrics over time;
- indicators related to churn, protocol events, and structural decentralization characteristics.

These endpoints allow consumers to understand the environment in which validators and pools operate, and to compare networks along structural dimensions rather than solely on the basis of individual actors.

Validators and pools

Validator and pool endpoints expose:

- static metadata (where available) and identifiers needed to link back to on-chain or external representations;
- time-series metrics such as participation, availability, penalties, configuration changes, stake share, and flows;
- derived indicators and factor scores that summarise aspects of reliability and behaviour.

Consumers can use these endpoints to build detailed views of individual validators and pools, to compare them within and across networks, and to analyse how their behaviour changes over time.

Providers

Provider endpoints aggregate information across validators and pools operated by the same entity:

- stake distribution and concentration across networks;
- aggregated reliability and operational indicators;
- exposure metrics for use in provider-level policies.

This enables policies and monitoring that target operational entities rather than only individual validators.

Events and alerts

Events and alerts represent structured signals about notable conditions observed by FortisX. API endpoints in this domain provide:

- access to raw or normalised events (for example, penalties, configuration changes, governance actions) tied to specific entities and times;
- alert streams indicating large stake movements, significant changes in performance, or deviations from expected patterns.

These outputs are suitable for integration into incident management, monitoring, and risk workflows.

Metrics and risk profiles

For more detailed analysis, the API exposes:

- specific metric series for entities over configurable windows and resolutions;
- risk profiles with factor scores and bucket classifications, tagged with model versions.

This allows consumers to incorporate FortisX factorisation of risk directly into their own logic, or to compare their own assessments with those calculated by the platform.

Policies and allocation evaluations

Finally, policy-related endpoints expose:

- the set of policy definitions active within a given scope, including metadata and versioning;
- evaluations of current allocations against policies, indicating where allocations are within bounds and where they diverge;
- allocation proposals generated by the policy engine, including target allocations, required changes, and references to the data and policies used.

These endpoints do not execute any actions. They bridge the gap between analytics and external execution systems, providing a structured representation of how FortisX would adjust allocations under given conditions and policies.

Integration patterns

The Analytics API is designed to support multiple styles of integration:

- **Periodic polling**

Systems that maintain their own internal state (for example, risk engines or reporting pipelines) can periodically query metrics, risk profiles, and allocation evaluations for their scope of interest.

- **Event-driven integration**

Using webhooks or message-based integrations, FortisX can deliver alerts and allocation proposals to downstream systems as they are generated, reducing the need for continuous polling.

- **Dashboard and tooling integration**

Internal tools and dashboards within an organisation can query the API directly to render network views, validator comparisons, risk summaries, or policy compliance reports.

- **Hybrid approaches**

For some use cases, a combination of scheduled queries for baseline data and event-driven delivery for changes or incidents is appropriate.

In all cases, the API is intended to be a source of structured data that can be combined with local context such as portfolio hierarchies, client-specific constraints, or internal approval records.

Versioning and compatibility

To maintain stability, the Analytics API follows an explicit versioning strategy:

- Endpoint families are versioned so that changes in response structure or semantics can be introduced without breaking existing integrations.
- Risk models and policy sets are versioned separately; API responses reference these versions, allowing consumers to log and interpret data under the correct assumptions.
- Deprecations are managed with notice periods and documentation, rather than abrupt removal of functionality.

Consumers are expected to:

- record model and policy version identifiers alongside the data they retrieve;
- migrate to new API versions according to their own change management processes.

This approach aligns with the broader emphasis of the platform on reproducibility and auditability.

Roadmap: evolution of analytics and interfaces

The FortisX platform is expected to evolve along several dimensions. While details and timelines are subject to change, the general direction of development can be grouped into a few themes.

Broader network coverage

Over time, additional validator-based networks may be integrated, provided they expose sufficient data to support FortisX analytics and policy logic. This includes:

- extending the ingest pipeline and data model to cover new protocols where appropriate;
- documenting network-specific nuances so that metrics and risk factors remain interpretable;
- ensuring that cross-network comparisons remain grounded in a coherent structure.

Network additions are evaluated based on technical characteristics, data availability, and their relevance to the organisations using the platform.

Deeper metrics and models

Within existing networks, analytics and risk modeling are expected to deepen:

- new metrics may be introduced as protocols evolve or as additional signals become available;
- factor definitions and model combinations may be refined based on observed behaviour and feedback from users;
- support for additional dimensions (for example, more granular infrastructure dependence indicators) may be added.

Such changes are introduced through the model versioning framework described earlier, with attention to compatibility and interpretability.

Policy expressiveness and tooling

The policy engine and configuration interfaces may be extended to:

- support richer constraints and combinations of conditions, within a structured and reviewable configuration model;
- provide simulation tools that allow users to test prospective policies against historical data;
- improve introspection into how specific clauses contributed to particular allocation evaluations or proposals.

The goal is to increase expressiveness without compromising determinism or transparency.

Integration and workflow support

To fit more closely into existing operational processes, additional integration and workflow features may be developed, such as:

- tighter integration patterns with common monitoring, incident management, or governance tooling;
- extended SDKs and client libraries in multiple languages that simplify API consumption;
- standardised data formats for exchanging allocation proposals and evaluations with execution systems.

These developments are focused on reducing integration friction rather than changing the core responsibilities of FortisX.

Operational and compliance artefacts

As the platform matures, FortisX may provide more structured artefacts for operational and compliance use, for example:

- enriched status and health endpoints for inclusion in external monitoring;
- extended logs and reports that summarise changes in policies, models, and allocation proposals over time;
- documentation that maps platform features to common control frameworks used by institutional users.

These artefacts are intended to help organisations align FortisX with their own governance and risk management practices.

Summary

The Analytics API is the external face of the FortisX analytics and policy engine. It:

- exposes the data model, metrics, risk profiles, policy evaluations, and allocation proposals in a structured, versioned form;
- supports multiple integration patterns, from dashboards to automated execution flows;
- is designed to evolve in a controlled way as network coverage, metrics, models, and policies develop.

The roadmap for FortisX emphasises incremental extension of coverage and depth, improved expressiveness and introspection in policies, and tighter integration with operational and compliance frameworks, while maintaining the core design principles of transparency, determinism, and separation of duties.