FINOS
Architecture as Code

2023-10-24

Say hi 👋

Please visit https://tinyurl.com/aasc-144 and add your name and company in a comment to the meeting issue

FINOS

Fintech
Open Source
Foundation

Antitrust Policy

All project meetings are subject to the Linux Foundation Antitrust Policy.
The following topics must not be discussed:

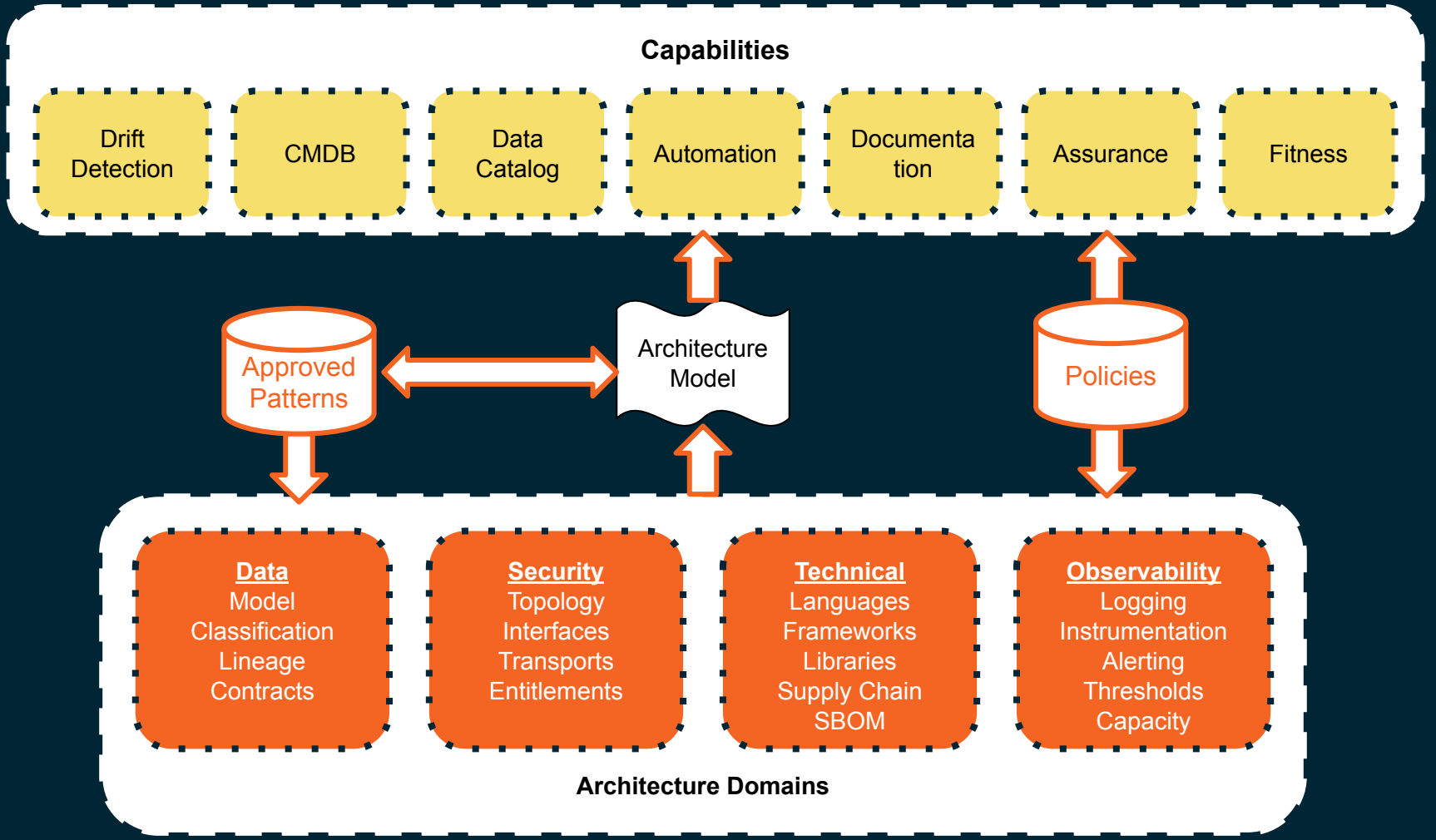
- Price-sensitive information
- Actual or projected changes in production, output, capacity or inventories
- Matters relating to bids, prospective bids, or bid policies
- Matters relating to actual or potential individual suppliers that might influence the business conduct of firms toward such suppliers
- Matters relating to actual or potential customers that might have the effect of influencing the business conduct of firms toward such customers
- Current or projected costs of procurement, development or manufacture of any product
- Market shares for any product or for all products
- Confidential or otherwise sensitive business plans or strategy

If you have questions, please contact legal@finos.org.

Previous Minutes

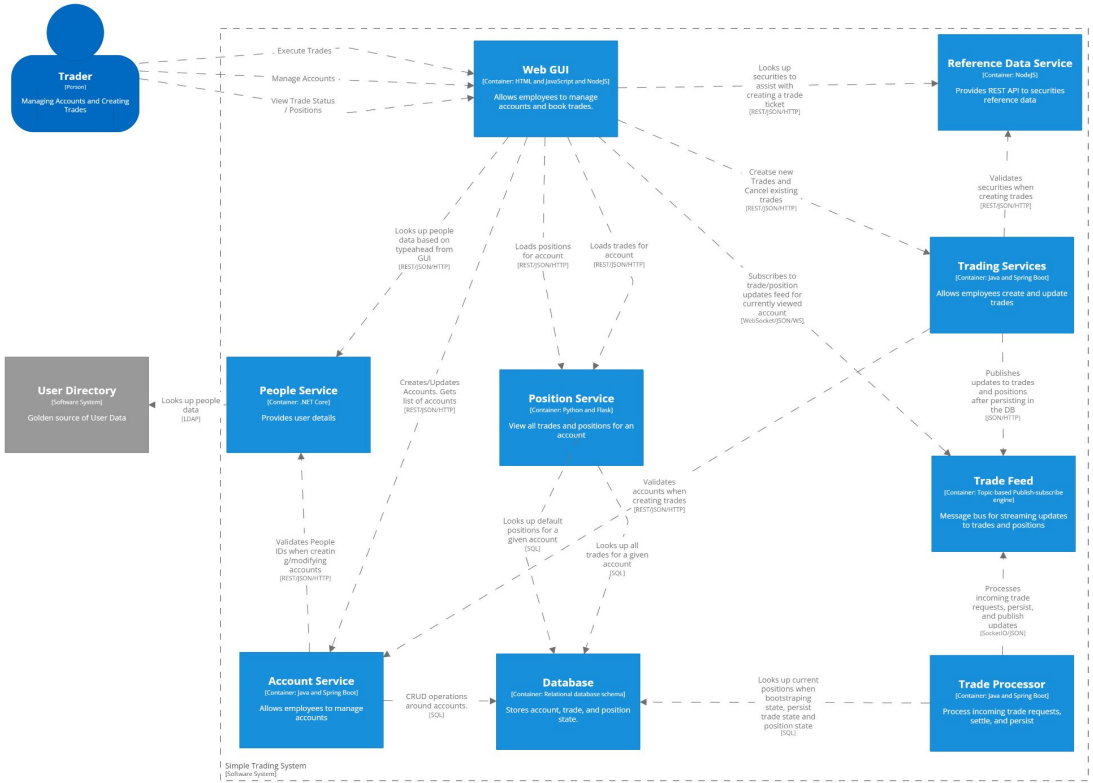
https://tinyurl.com/aasc-138

Logical Architecture as Code Model - Recap


Capabilities
Drift Detection
CMDB
Data Catalog
Automation
Documentation
Assurance
Fitness
Approved Patterns
Architecture Model
Policies
Data
Model
Classification
Lineage
Contracts
Security
Topology
Interfaces
Transports
Entitlements
Technical
Languages
Frameworks
Libraries
Supply Chain
SBOM
Observability
Logging
Instrumentation
Alerting
Thresholds
Capacity
Architecture Domains

Core
Manifest Walkthrough

An Example: TraderX



Nodes


```
{
 "uniqueId": "traderx-trader",
 "type": "actor",
 "name": "Trader",
 "description": "Person who manages accounts and executes trades"
}
```

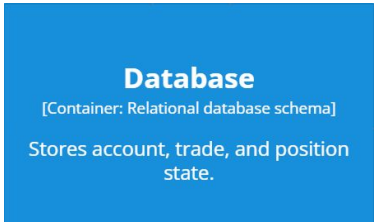
Actors

Nodes


```
{
 "uniqueId": "web-client",
 "type": "webclient",
 "name": "Web  Client",
 "description": "Browser based web interface for TraderX",
 "data-classification": "Confidential",
 "run-as": "user"
}
```


Processes

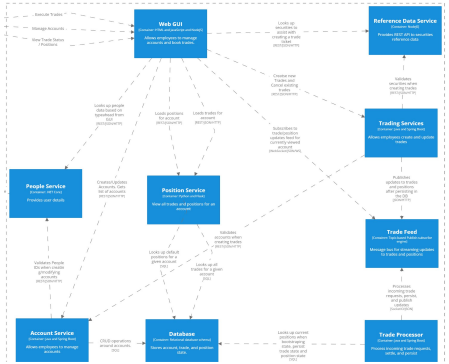
Nodes




```
{
 "uniqueId": "traderx-db",
 "type": "database",
 "name": "TraderX DB",
 "description": "Database which stores account, trade and position state",
 "data-classification": "Confidential",
 "run-as": "systemId"
}
```


Components

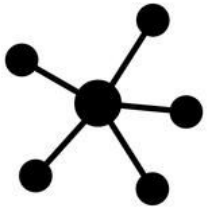
Nodes




```
{
 "uniqueId": "traderx-system",
 "type": "system",
 "name": "TraderX",
 "description": "Simple Trading System"
}
```

System

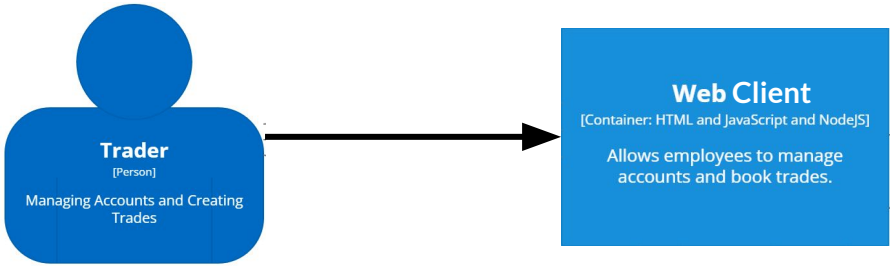
Nodes




```
{
 "uniqueId": "internal-bank-network",
 "type": "internal-network",
 "name": "Bank ABC Internal Network",
 "description": "Internal network for Bank ABC",
 "instance": "Internal Network"
}
```


Network

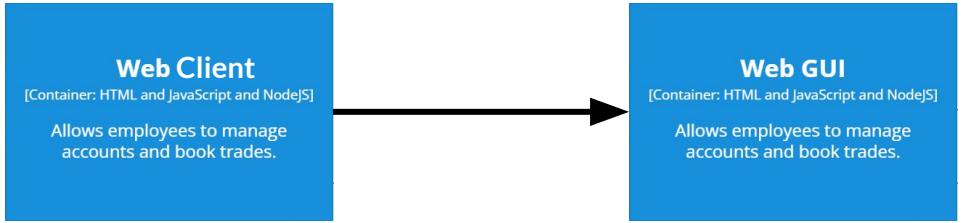
Relationships




```
{
 "uniqueId": "trader-uses-web-client",
 "type": "interacts",
 "parties": {
   "actor": "traderx-trader",
   "nodes": [
     "web-client"
   ]
 }
}
```

Interacts

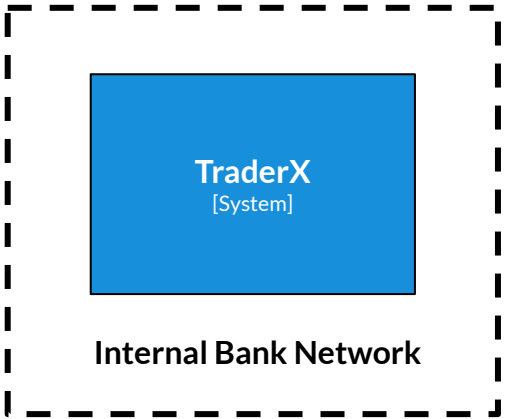
Relationships




```
{
 "uniqueId":
"web-client-uses-web-gui",
 "type": "connects",
 "parties": {
   "source": "web-client",
   "destination": "web-gui-process"
 },
 "protocol": "HTTPS",
 "authentication":
"SiteMinder/Isolated"
}
```


Connects

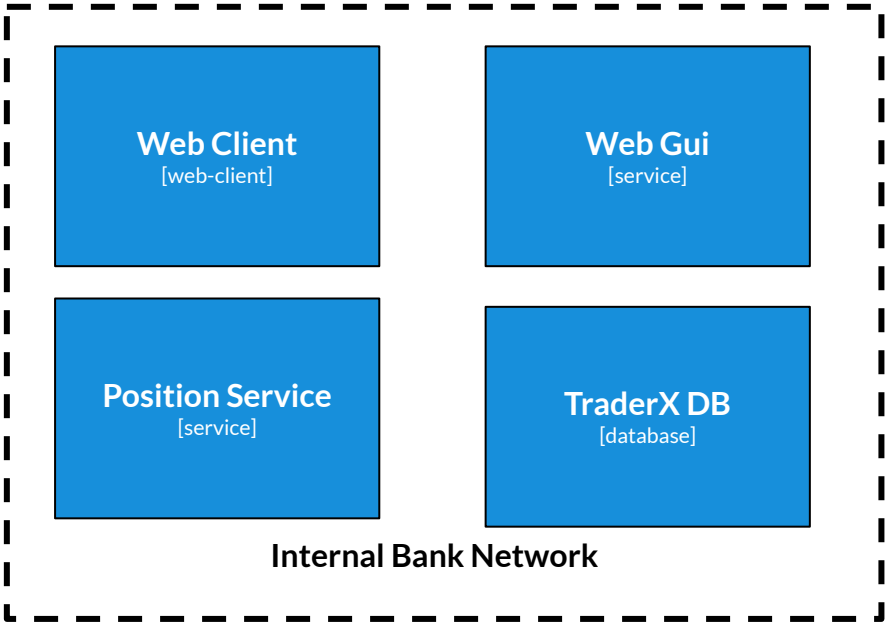
Relationships




```
{
 "uniqueId":
"traderx-system-is-deployed-in-interna
l-bank-network",
 "type": "deployed-in",
 "parties": {
   "nodes": [
     "traderx-system"
   ],
   "container":
"internal-bank-network"
 }
}
```

Deployed-In

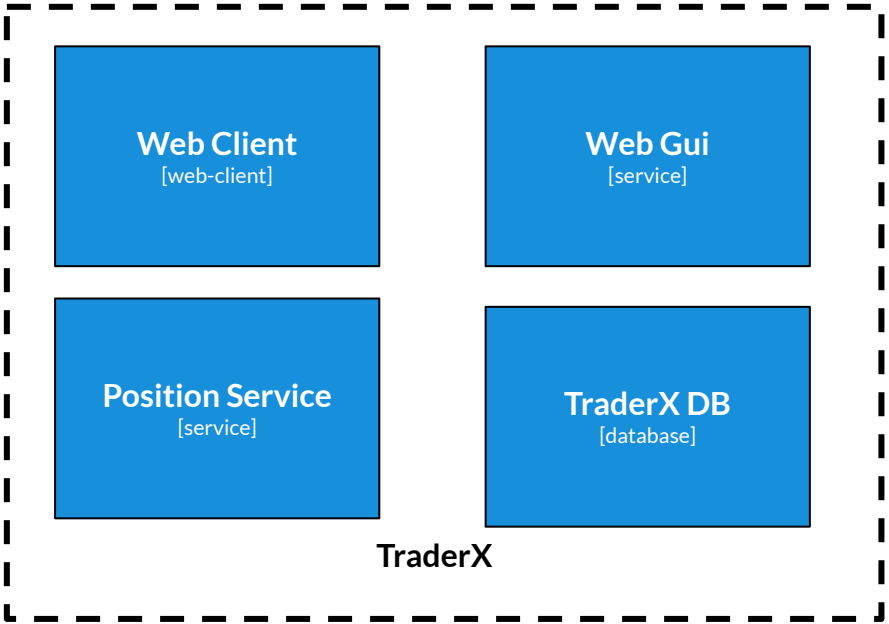
Relationships




```
{
 "uniqueId":
"traderx-system-components-are-deploye
d-in-internal-bank-network",
 "type": "deployed-in",
 "parties": {
   "nodes": [
     "web-client",
     "web-gui-process",
     "position-service",
     "traderx-db"
   ],
   "container":
"internal-bank-network"
 }
}
```

Deployed-In

Relationships



Composed-Of

```
{
 "uniqueId":
"traderx-system-is-composed-of",
 "type": "composed-of",
 "parties": {
   "nodes": [
     "web-client",
     "web-gui-process",
     "position-service",
     "traderx-db"
   ],
   "container": "traderx-system"
 }
}
```

Nodes capture capability

Relationships provide multi-dimensional views across your architecture

How can we overlay different architectural aspects?

DOMAINS

Domains

A clean logical model with strongly defined domain specific adaptations

Potential Domains

- Security
- Data
- Deployment
- Cross-Functional
- Business
- Technical (SBOM)
- Asset Inventory

Security Domain

Identify risks and control vulnerabilities

Risk: Unauthorised Access

Control: Firewall Rule

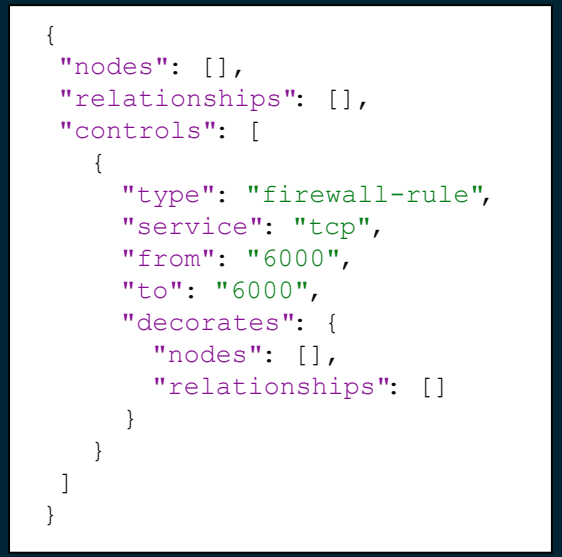
Risk: Cross Site Request Forgery (CSRF)

Control: Non-predictable token

Domains add context

Each domain provides a **supplemental schema definition**

e.g.
Security adds a controls[ ] array to the existing nodes[ ] and relationships [ ]



Domain

Extension

Domains are official extensions of the specification that solve wide ranging use cases

Extensions enable user specific cases to be solved or enable capabilities

Extensions with broader applicability should be proposed for promotion as domains

On-site Accelerator

In-Person - London, October 30th

Please indicate interest to attend via the meeting issue