Z Schema Description

Formal Specification for Client Identifying Data Requirements for Banks in Switzerland
(Requirements published by the Swiss Financial Authority, or FINMA, S. References)

Specification Written in Z Notation

Developed by Serge (Siarhei Vinahradau, vinahradau@yahoo.de)

Contents

Z Schema Description....1
Developed Schema....2
References....2
FINMA Specification Requirements and their Equivalents in Z....3
Data Classification....5
Implementation of Data Classification....6
Data Storage....7
Roles, Users and Access Rights....9
Data Access....11
Bulk Data Access....12

Developed Schema

Z Schema (edited with CZT plugin)
https://github.com/vinahradau/finma/blob/master/CIDFINMA_spec_Z.zed16

Z Schema in Latex (can be run by Jaza Animator)
https://github.com/vinahradau/finma/blob/master/CIDFINMA_spec_Z.zed

Test Cases and Jaza Scripts
https://github.com/vinahradau/finma/tree/master/jaza

Alloy
https://github.com/vinahradau/finma/blob/master/cid_domain_specification_alloy.als

References

FINMA-Rundschreiben „Operationelle Risiken – Banken" (published in 20.11.2008, last updated in 2017). https://www.finma.ch/de/~/media/finma/dokumente/rundschreiben-archiv/finma-rs-200821---30-06-2017.pdf

CZT Development Environment for Z
http://czt.sourceforge.net/

Jaza Animator for Z
https://github.com/uho/jaza/blob/master/README.txt

FINMA Specification Requirements and their Equivalents in Z		
#	**Natural language definition / FINMA ref**	**Z notation equivalent**
1	Client Identification Data has to be classified as: – direct, e.g. name; – indirect, e.g. passport number; – potentially indirect, e.g. combinations of DOB, occupation, citizenship etc. (FINMA 10*). –	DATACATEGORY ::= DIRECT | INDIRECT | POTENTIALLYDIRECT | PROTECTED | NONCID CIDCATEGORIES == {DIRECT, INDIRECT, POTENTIALLYDIRECT} dataClassification: METADATA ⇸ DATACATEGORY
2	Higher customer protection risks of storing and accessing data outside Switzerland: data has to be protected, i.e. anonymized, pseudoanonymized or encrypted (FINMA 20*)	DATACATEGORY ::= PROTECTED CONTENT ::= XXXXX COUNTRY ::= SWITZERLAND | UK | USA | GERMANY nodeCountry = SWITZERLAND ∨ (∀ c : ran nodeDataCategories • c ∉ CIDCATEGORIES)
3	CID data owners are responsible for CID data classification and the whole data life cycle (FINMA 13*)	DataOwner: METADATA ⇸ ENTITY ImplementDataClassification == AssignDataOwner ∧ ClassifyDataCategory RecycleData
4	There must be a role and function based system of users' access rights in place (FINMA 22*)	ROLE ::= ROLEGUICIDUSER | ROLEGUIUSER | ROLEBULKCID | ROLEBULK | ROLE1 roles: ROLE ↔ METADATA userAccessRigths: USER ↔ ROLE accessNodeMetadata? ∈ roles⦇(userAccessRigths⦇{user?}⦈)⦈
5	An internal employee has to be responsible for the compliance of outsourced CID activities (FINMA 50*)	teams: ENTITY ↔ USER internalUsers: ℙ USER externalUsers: ℙ USER ∀ u : externalUsers • ¬(userAccessRigths⦇{u}⦈ ∩ CIDROLES ≠ ∅ ∧ teams⦇dom (teams ▷ {u})⦈ ∩ internalUsers = ∅)
6	There must be an inventory of CID storage locations, processing applications, access locations, access applications (FINMA 15*)	cidStoringNodesIds: ℙ NODEID cidStoringNodesIds′ = cidStoringNodesIds ∪ {nodeIdInput?}

7	CID data accessed from outside Switzerland has to be protected (FINMA 20*)	nodeDataCategories⦇{accessNodeMetadata?}⦈ ⊆ CIDCATEGORIES ∧ userCountry? ≠ SWITZERLAND ∧ contentOutput! = {XXXXX})
8	Bulk access to CID by entitled users must be recorded, e.g. in log files (FINMA 40*)	cidBulkAccess: USER ↔ NODEID cidBulkAccess′ = cidBulkAccess ∩ {(user?, nodeId?)}
9	List of users with bulk CID access (FINMA 34*)	BulkCIDAccessUsersList BulkCIDAccessUsersList! = dom (userAccessRigths ▷ {ROLEBULKCID})

Data Classification

DATACATEGORY

CIDCATEGORIES

According to FINMA 10*, categories of client identifying data (CID) include:

- direct (e.g. customer name): DIRECT;
- indirect (e.g. passport number): INDIRECT;
- potentially indirect (combination of data): POTENTIALLYINDIRECT.

In our specification we also use:

- NONCID (for data not classified as CID).
- PROTECTED (for protected, e.g. anonymized CID data, in which case it's not CID anymore).

DIRECT, INDIRECT and POTENTIALLYINDIRECT is included into the CIDCATEGORIES collection accordingly.

```
DATACATEGORY ::= DIRECT | INDIRECT | POTENTIALLYDIRECT | PROTECTED | NONCID
CIDCATEGORIES == {DIRECT, INDIRECT, POTENTIALLYDIRECT}
```

Implementation of Data Classification

METADATA
CONTENT
ENTITY
DOMAIN

InitDomain
AssignDataOwner
ClassifyDataCategory
ImplementDataClassification
RecycleData

In our schema, METADATA is an abstraction for data descriptors or attributes, such as CUSTOMERNAME, CUSTOMERADDRESS or ISVIPCUSTOMER (in reality, it can be a technical data descriptor, e.g. a database schema, table and column or an XML attribute path).

We describe the link between METADATA and DATACATEGORY, understood as CID data classification, via a partial function in DOMAIN schema. Here the function imposes its own constraint: a domain element of dataClassification, which is of type METADATA, can be linked to at most one data category.

dataClassification: METADATA ⇸ DATACATEGORY

Data owners (ENTITY in our schema) are responsible for the whole data lifecycle, including the CID data classification (FINMA 14*).

dataOwner: METADATA ⇸ ENTITY

The following constraint in DOMAIN schema states that domain of dataClassification function is a subset of dataOwner, i.e. the data classification happens after the data ownership assignment:

dom dataClassification ⊆ dom dataOwner

Operations AssignDataOwner and ClassifyDataCategory populate the functions, and ImplementDataClassification is the sum of the two operations above. RecycleData does the opposite, removing the links.

Data setup example:

dataOwner == {(CUSTOMERNAME, ENTITY1), (ISVIPCUSTOMER, ENTITY1)}
dataClassification == {(CUSTOMERNAME, DIRECT),(ISVIPCUSTOMER, NONCID)}

Data Storage

COUNTRY
CONTENT
NODEID
NODE
CIDSTORINGNODESAUDITLOG

AddNodeData

NODE is an abstraction for systems and applications (FINMA 15*).

Our NODE schema has its own data category classification function:

$\mathrm{nodeDataCategories} : \mathrm{METADATA} \nrightarrow \mathrm{DATACATEGORY}$

If CID data is stored outside Switzerland, it has to be protected, e.g. anonymized (FINMA 20*). The NODE schema has a corresponding constraint:

$\mathrm{nodeCountry} = \mathrm{SWITZERLAND} \lor (\forall c : \mathrm{ran}\ \mathrm{nodeDataCategories} \bullet c \notin \mathrm{CIDCATEGORIES})$

As already mentioned, in cases of CID data protection (e.g. anonymization), data loses its CID status (in our schema, PROTECTED is not a member of CIDCATEGORIES). As a consequence, nodeDataCategories may differ from the global dataClassification: the same data may be classified by the data owner as CID, but stored as protected (non-CID) on a node.

Data contents (or data values) are described as nodeDataContents function. Our schema expects that for all stored data contents, their data categories are known (NODE schema):

$\mathrm{nodeDataContents} : \mathrm{METADATA} \nrightarrow \mathrm{CONTENT}$
$\mathrm{dom}\ \mathrm{nodeDataContents} \subseteq \mathrm{dom}\ \mathrm{nodeDataCategories}$

The bank has to know where CID data is stored (FINMA 15*). In our schema, cidStoringNodesIds is a power set of node identifiers, and the following constraint expects identifiers of all nodes with stored CID data to be recorded in cidStoringNodesIds:

$\mathrm{cidStoringNodesIds} : \mathbb{P}\ \mathrm{NODEID}$
$\forall \mathrm{cidDataCategory} : \mathrm{ran}\ \mathrm{nodeDataCategories} \bullet$
$\mathrm{cidDataCategory} \in \mathrm{CIDCATEGORIES}$
$\Rightarrow$
$\mathrm{nodeId} \in \mathrm{cidStoringNodesIds}$

Data storage logic is described in AddNodeData operation:

- Case 1: non-CID; store data;
- Case 2: CID and node in Switzerland; store data; create a node identifier entry in cidStoringNodesIds;
- Case 3: CID and node not in Switzerland ; store protected (anonymized) version of the data (XXXXX is stored instead of the data content coming from nodeDataContentInput?).

Case 3:

$nodeDataContents' = nodeDataContents \oplus \{nodeMetadataInput? \mapsto XXXXX\}$
$\wedge$
$nodeDataCategories' = nodeDataCategories \oplus \{nodeMetadataInput? \mapsto PROTECTED\})$

Stored data example, node in Switzerland:

nodeDataCategories=={(CUSTOMERNAME, DIRECT), (ISVIPCUSTOMER, NONCID)}
nodeDataContents == {(CUSTOMERNAME, MUSTERMANN),
(ISVIPCUSTOMER, YES)}

Stored data example, node outside Switzerland:

nodeDataCategories=={(CUSTOMERNAME, PROTECTED),
(ISVIPCUSTOMER, NONCID)}
nodeDataContents=={(CUSTOMERNAME, XXXXX), (ISVIPCUSTOMER, YES)}

Roles, Users and Access Rights

ROLE
USER
ENTITY

AddRol
AddUse
AddInternalUse
AddExternalUse
AddUserAccessRight
BulkCIDAccessUsersList

To regulate access to CID data, Finma expects banks to have a role or function based authorisation system (FINMA 22*). To describe the authorization system, our schema introduces the abstractions of ROLE and USER. Relations are defined as follows:

$\text{roles}: \text{ROLE} \leftrightarrow \text{METADATA}$
$\text{userAccessRigths}: \text{USER} \leftrightarrow \text{ROLE}$

For the data structures above, we choose relations, not functions, since a role can be linked to multiple METADATA instances, and USER can have multiple roles.

Thus, USER is not assigned access to METADATA instances directly, but rather through ROLE. This approach is an example of RBAC Access Control Model (Role Based Centralized) and provides additional flexibility in access rights maintenance.

To populate roles and userAccessRigths relations, AddRole and AddUserAccessRights operations use mathematical unions of the existing relations and new pairs created from the inputs:
$\text{roles}' = \text{roles} \cup \{(\text{role?}, \text{metadata?})\}$
$\text{userAccessRigths}' = \text{userAccessRigths} \cup \{(\text{user?}, \text{role?})\}$

Authorization data example:
roles=={(ROLEGUIUSER, CUSTOMERNAME), (ROLEGUIUSER, ISVIPCUSTOMER)}
userAccessRigths=={(USER1, ROLEGUIUSER)}

FINMA 50* expects internal employees to supervise outsourced CID activities. To express this constraint, the model distinguishes between internal and external users:
$\text{internalUsers}: \mathbb{P}\,\text{USER}$
$\text{externalUsers}: \mathbb{P}\,\text{USER}$

All users are also assigned a team:
$\text{teams}: \text{ENTITY} \leftrightarrow \text{USER}$

The invariant states that no external users have CID roles and no internal users (supervisors) in their teams:
$\forall\, u : \text{externalUsers} \bullet$
$\neg(\text{userAccessRigths}(\!|\{u\}|\!) \cap \text{CIDROLES} \neq \emptyset$
$\land$
$\text{teams}(\!|\text{dom}\,(\text{teams} \rhd \{u\})|\!) \cap \text{internalUsers} = \emptyset)$

FINMA 40* expects a bank to maintain a list of all users with bulk CID access rights. This requirement is implemented in operation $\mathsf{BulkCIDAccessUsersList}$. The operation uses the range restriction ($\rhd$) to look up in a relation all tuples with the CID bulk access role in their values, and then takes the domicile (key) of the tuples. The result is a set of users.

$$\mathsf{userAccessRigths}: \mathsf{USER} \leftrightarrow \mathsf{ROLE}$$

$$\mathsf{BulkCIDAccessUsersList!} = \mathsf{dom}\,(\mathsf{userAccessRigths} \rhd \{\mathsf{ROLEBULKCID}\})$$

Data Access

AccessNode

Assuming the necessary access rights are available, a user can access particular data stored on a node. This logic is described in AccessNode operation.

- Case 1: user tries to access CID data from outside Switzerland: a protected (anonymized) version of the data is transmitted (instead of the real data content, XXXXX is stored in the contentOutput! Variable);
- Case 2: otherwise (non-CID or user in Switzerland); transmit the unmodified data to the user (unchanged data content is stored in contentOutput! Variable).

Thus, requirement in FINMA 20* is fulfilled.

Data access output example, CID data for a user in Switzerland:

contentOutput!=={MUSTERMANN}

Data access output example, CID data for a user outside Switzerland:

contentOutput!=={XXXXX}

Bulk Data Access

CIDBULKLOG

AccessBulk

Bulk access is a process of accessing big quantities of data (FINMA 40*). For CID bulk access, the FINMA regulation expects a special procedure, e.g. instances of bulk access being documented in log files. As an abstraction for bulk access logs, our schema defines a relation between users and nodes. Here, a user may be a part of multiple log entries (pairs), hence a relation, not a function.

$cidBulkAccess: USER \leftrightarrow NODEID$

The bulk access logic is described in AccessBulk operation. Only 2 cases are defined here:

- Case 1: user in Switzerland accessing a node with stored CID data: all node data is stored in contentOutput! , and a cidBulkAccess log entry is created;
- Case 2: otherwise, all node data is stored in contentOutput! , and no cidBulkAccess log entry is created.

Mathematically, the absence of CID data on a node is described in AccessBulk operation in the following predicate: no data categories stored on a node belong to the set of CID data categories (their intersection is empty).

$\mathrm{ran}\ nodeDataCategories \cap CIDCATEGORIES = \varnothing$

For the bulk access to be enabled, our schema does not expect AddRole to be called, since AccessBulk should provide access to all data on a node, regardless of the metadata descriptors. Thus, the authorization logic in AccessBulk does not check the metadata descriptor of the data to be transmitted. Here in AccessBulk operation, checking ROLEBULKCID and ROLEBULK roles of the user is sufficient, as an opposite to AccessNode operation, where the metadata descriptors of the stored data are explicitly checked.

Bulk data access example, CID data and non-CID data, user inside Switzerland:
contentOutput!=={MUSTERMANN, YES}

Bulk data access example, protected data and non-CID data, user inside or outside Switzerland:
contentOutput!=={XXXXX, YES}