

Naming – thoughts on refactoring

Table of Contents

Abstract.....	1
Motivation.....	2
Introduction to Naming data landscape.....	3
Visualization (1) – ESS names and name part structure.....	4
Visualization (2) – name and name part structure, tables, objects.....	4
Visualization (3) – database, tables, subset of columns.....	5
Visualization (4) – user, application, database, tables, objects.....	6
Exploring Options.....	6
Suggestions.....	7
Proof-of-Concept.....	8
Way forward.....	8
Thoughts.....	9
Conclusion.....	9
References.....	9
Application.....	9
Repository.....	9
Appendix.....	10
A. Database scripts.....	10
A1. Current database script.....	10
A2. Suggestion 1, database script, includes migration of data.....	10
A3. Suggestion 2, database script, includes migration of data.....	10
A4. Suggestion 3, database script.....	10
A5. Suggestion 4, database script, includes migration of data.....	10
B. Database visualizations.....	10
B1. Database visualizations & Proof-of-concept.....	10
B2. Persistence Layer.....	11

Abstract

Naming data model and storage is complex and difficult to understand and maintain which makes Naming application complex and difficult to understand and maintain. This paper explores current data model and storage and application, together with options for refactored data model and storage and outline of refactored application. Different sections in paper explores various aspects of data model and storage and application, in various detail.

Note that implementation of application is not explored. It is sufficed to know that it may be done.

Motivation

Naming application is stable but difficult to understand. This makes for long time to understand enough before enough knowledge is acquired to work comfortably with application and database. What you see in application is not what you get from database. There is a distance between data storage and data presentation which manifests itself in a number of steps, model objects, business objects, interpretation of business objects, caching of business objects.

Some factors make for more difficulties than other factors. Model for data storage does not correspond to reality but rather a compact way of storing hierarchical data in few tables with help of recursive relations. Entire content of Naming database is kept in two caches to allow for normal responsiveness in application.

Caches

1. names together with name part structures. Used in UI. *tree structure of business objects*
2. names. Used in REST API and import from Excel. *flat structure of business objects*

When there is change for names or name part structure, database is re-read, content interpreted into business objects and caches re-read. Parallel with caches, database is also used, e.g. for history operations. Thus, UI and REST API use combinations of caches and database queries.

Database design is flexible but complex. Name part structure of arbitrary level may be handled. This is handled through recursive references. As a result, caching was introduced in order to have a usable application for existing database. Modification of data is handled as revisions of data which in practice translates to rows in tables. Amount of data is in order of 1000+ entries for name part structure and in order of 100000+ entries for names (includes history).

A different data model and storage, with closer resemblance to names and name part structure as in Naming convention, allows for easier understanding. In other words, a data model closer to domain model forms a better base for understanding than today. This in turn makes for less time before enough knowledge is acquired to work comfortably with application and database. Moreover, and more importantly, this allows for removal of caches.

Refactoring of data model and storage, and removal of caches, require refactoring of application.

A more straightforward approach for database model and storage allows for better understanding, less distance between storage and usage and easier modification when handling requests for changes. It also allows for more and better usage of REST API. Such usage is currently done by a number of known applications and systems, e.g. CCDB ecosystem (Cable, CCDB, IOC Factory), CHESS, IOC deployment tool, Awesome Naming Tool, Device viewer, PV Validator. On top of this, REST API is also used by Infrastructure group for monitoring of applications. Since REST API has no authentication or token, usage is at least as mentioned but may be greater.

As REST API usage is foreseen to increase, care is to be taken to aid its development and usage.

Introduction to Naming data landscape

Naming Tool manages the naming of ESS wide physical and logical devices according to ESS Naming Convention.

Naming consist of names (ESS names) and name part structures (System Structure, Device Structure). Name part structures are hierarchical and consist of 1-3 layers.

System Structure layers

Which part of the facility does the device provide service to?

1	System Group	may contain System
2	System	belongs to System Group, may contain Subsystem
3	Subsystem	belongs to System

Device Structure layers

What kind of service does the device provide?

1	Discipline	may contain Device Group
2	Device Group	belongs to Discipline, may contain Device Type
3	Device Type	belongs to Device Group

ESS name

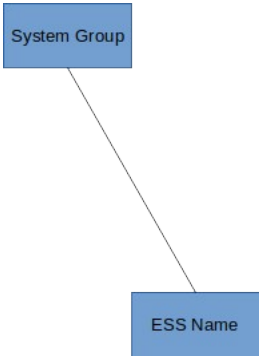
must refer to System Structure layer
may refer to Device Structure layer 3
may have index for instance of name (to avoid potential duplicates)

There are rules for names, mnemonics, descriptions, and other fields. An name part entry must have mnemonic to be part of ESS name. Each entry in either System Structure, Device Structure, ESS name has identifier UUID. This is unique for entry and remains same across lifespan of data (revisions). Additional data such as who did what and when are also stored.

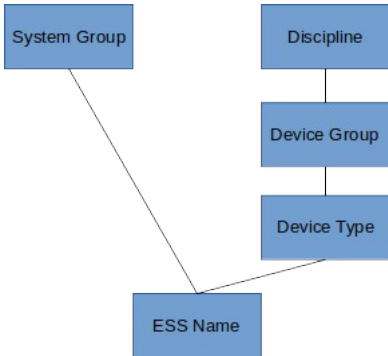
For current implementation, revisions for all levels of System Structure and Device Structure are stored in same table through references to parent that may have revisions that in turn references parent that in turn may have revisions. Revisions for ESS name are stored in same table through references to System Structure and Device Structure. In order to be useful for application, revisions are interpreted into business objects.

Revision handling is important to consider current information “where we are” and “where we are going”.

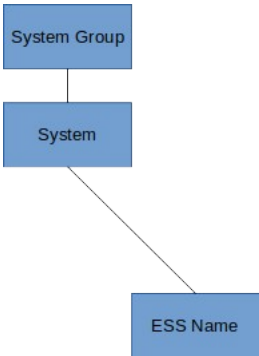
Visualization (1) – ESS names and name part structure



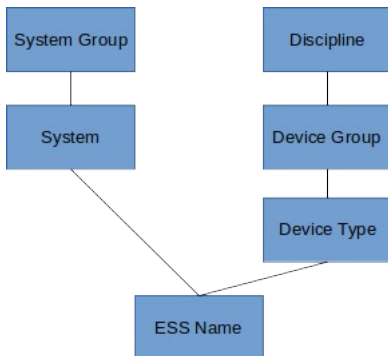
Acc



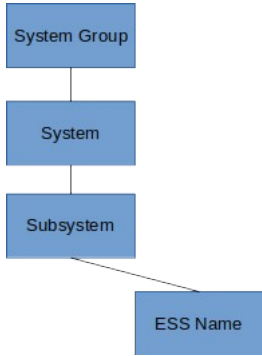
Acc:RFS-PrlTap-054



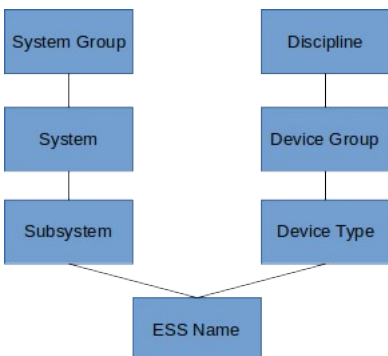
A2T



A2T:RFS-PrlTap-054

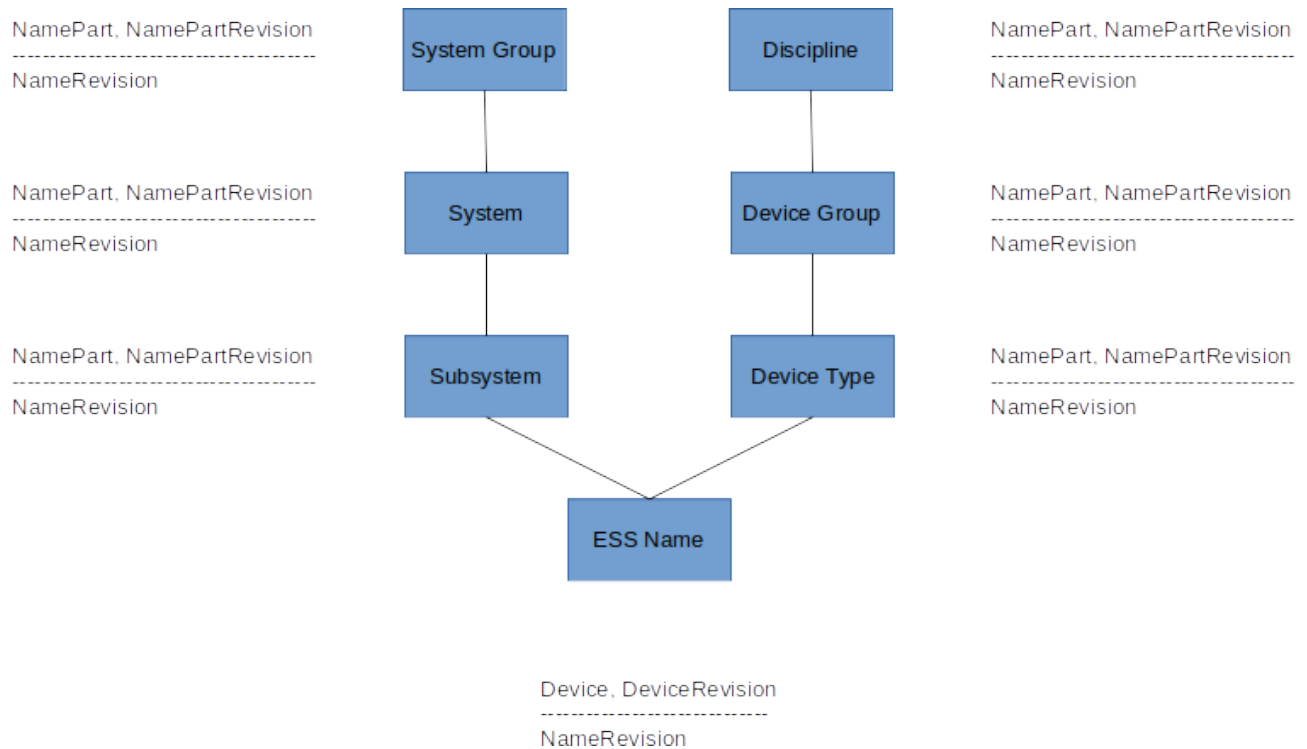


A2T-010PRL



A2T-010PRL:RFS-PrlTap-054

Visualization (2) – name and name part structure, tables, objects



Visualization (3) – database, tables, subset of columns

System structure

namepart, namepartrevision

id	id	version	deleted	mnemonic	name	status	namepart_id	parent_id
1	1	4	false	Acc	Accelerator	APPROVED	1	[NULL]
1	1050	1	false	[NULL]	Accelerator	APPROVED	1	[NULL]
16	16	2	false	A2T	Accelerator to Target	APPROVED	16	1
16	1708	1	false	A2T	Accelerator to Target	APPROVED	16	1
1359	2292	3	false	010PRL	01 Phase Reference Line	APPROVED	1359	16

Device structure

namepart, namepartrevision

id	id	version	deleted	mnemonic	name	status	namepart_id	parent_id
242	242	2	false	RFS	RF Systems	APPROVED	242	[NULL]
1349	2282	1	false	[NULL]	Phase Reference Line	APPROVED	1349	242
1350	2692	3	false	PRLTap	Phase Reference Line Tap	APPROVED	1350	1349

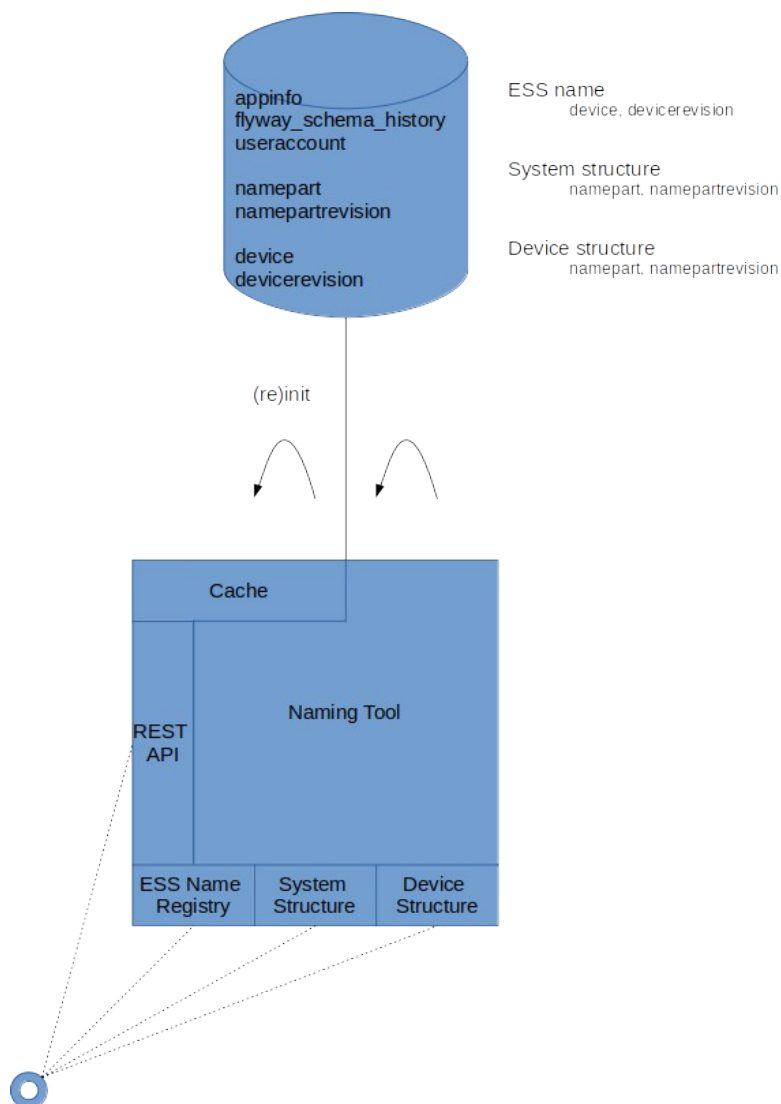
ESS name

device, devicerevision

id	id	conventionname	conventionname eqclass	deleted	instanceindex	device_id	devicetype_id	section_id
3957	32540	A2T-010PRL:RFS-PRLTap-054	A2T-10PR1:RFS-PR1TAP-54	false	054	3957	1350	1359

There may be multiple revisions for System Structure and Device Structure entries and also for ESS name entries. There is example of ESS name with 17 revisions which correspond to 17 rows in devicerevision table. In above examples are multiple revisions (two each) for System Structure entries Accelerator and Accelerator to Target.

Visualization (4) – user, application, database, tables, objects



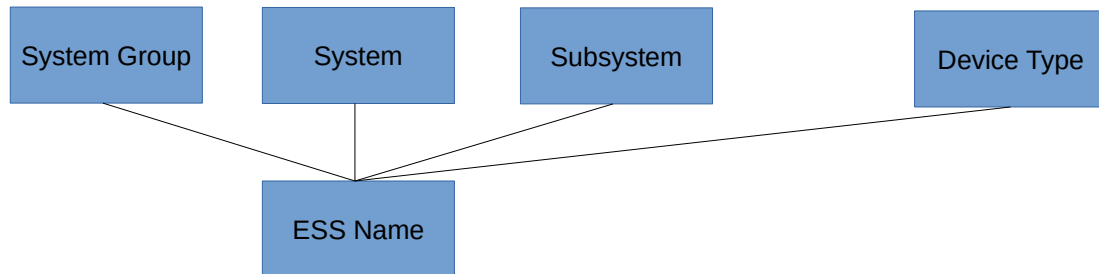
The circle in lower left is user interacting with Naming application.

Exploring Options

A data model and storage that is easier to understand is desired.

This may be manifested in a flatter data model and storage.

Outline of name and name part structure



ESS Name must refer either of System Group, System, Subsystem.

ESS Name may refer Device Type.

There is table for ESS name that refers to tables for System Group, System, Subsystem (System Structure) and Device Type (Device Structure).

id	name	systemgroup_id	system_id	subsystem_id	devicetype_id	instanceindex
		(optional)	(optional)	(optional)	(optional)	(optional)

Together with above are tables for name part structures.

- tables for System Group, System, Subsystem (System Structure)
- tables for Discipline, Device Group, Device Type (Device Structure)

Each table

id	name	mnemonic

Additional data such as who did what and when are also stored. There may be multiple revisions. Revision handling is important to consider current information “where we are” and “where we are going”.

See appendix for detailed suggestions, database and script.

Suggestions

	Suggestion 1	Suggestion 2
	keep uuid in existing tables	transfer uuid of namepart, device to new tables
keep tables	<ul style="list-style-type: none">• appinfo• flyway_schema_history• user_notification• namepart• device	<ul style="list-style-type: none">• appinfo• flyway_schema_history• user_notification
remove tables	<ul style="list-style-type: none">• namepartrevision• devicerevision• useraccount	<ul style="list-style-type: none">• namepart• namepartrevision• device• devicerevision• useaccount
new tables	<p>System Structure</p> <ul style="list-style-type: none">• namepartrevision_systemgroup• namepartrevision_system• namepartrevision_subsystem <p>Device Structure</p> <ul style="list-style-type: none">• namepartrevision_discipline• namepartrevision_devicegroup• namepartrevision_devicetype <p>ESS name</p> <ul style="list-style-type: none">• devicerevision_device	<p>System Structure</p> <ul style="list-style-type: none">• namepartrevision_systemgroup• namepartrevision_system• namepartrevision_subsystem <p>Device Structure</p> <ul style="list-style-type: none">• namepartrevision_discipline• namepartrevision_devicegroup• namepartrevision_devicetype <p>ESS name</p> <ul style="list-style-type: none">• devicerevision_device

Table useraccount with user and role is not clear as role may change over time. Instead of revisions referring to useraccount table, username may be written in revision. (*to be investigated*)

A key part of refactoring data model and storage is keeping existing content with history.

Suggestion 1 is somewhat more straightforward for database scripts as existing namepart and device tables are kept.

Proof-of-Concept

Proof-of-concept is done by considering that REST API should stay the same and how implementation may be updated to handle suggestions. Proof-of-concept is mostly a theoretical exercise, e.g. that REST API endpoint may be handled by having implementation query certain tables in certain order.

See Appendix B1 - Database visualizations & Proof-of-concept.

Way forward

A lot of things need to be considered when going forward with refactoring of Naming.

Among them are:

- Lazy loading (no lazy loading for primefaces tree component)
- ORM, Hibernate, eager vs. lazy

- ORM, Hibernate, no regex
- Performance
- Regex
- REST API implementation – compare with today
- UI, stay same or not, similar or not, depend on REST API or not

Thoughts

Current database table allows tree hierarchy of arbitrary depth through recursive references, thereby having flexible approach. However hierarchy is complex and difficult to understand.

A flatter data model and storage is easier to understand and maintain than a tree hierarchy with recursive references. The same applies for application that uses data model and storage.

Moreover, a flatter data model and storage allows for database queries to retrieve information and solve tasks in application UI and REST API without having to maintain multiple caches for names and name structures.

Conclusion

It is desired to have a data model and storage, with closer resemblance to domain model (names and name part structure) than today. This makes for less time before enough knowledge is acquired to work comfortably with application and database.

Future usage of Naming can be predicted to some extent. This includes current usage as in application UI and REST API. As more usage of REST API is foreseen, it is important to support retrieval (and possible storing) of information (minimum not block options).

Refactoring of data model and storage require refactoring of application.

Recommendation is to refactor Naming data model and storage and application.

References

Application

- <https://naming.esss.lu.se/>
- <https://naming.esss.lu.se/rest/>

Repository

- <https://gitlab.esss.lu.se/ics-software/naming-convention-tool>
 - master branch
 - documentation/developer/application
 - naming_accelerator_as_system.pdf
 - naming_developer_documentation.pdf

Appendix

A. Database scripts

A1. Current database script

See file(s)

- Appendix_A1
 - V1__Initial.sql
 - V2__Commit_Msg_to_Device.sql
 - V3__Notification_CC_List.sql

A2. Suggestion 1, database script, includes migration of data

See file(s)

- Appendix_A2
 - script_notes_2_id.txt

A3. Suggestion 2, database script, includes migration of data

See file(s)

- Appendix_A3
 - script_notes_3.0_uuid.txt
 - script_notes_3.1_uuid.txt (supersedes 3.0)

A4. Suggestion 3, database script

Not (yet) available

A5. Suggestion 4, database script, includes migration of data

B. Database visualizations

B1. Database visualizations & Proof-of-concept

See file(s)

- Appendix_B1
 - database_diagrams.pdf
 - Naming database
 - current
 - Suggestion 1
 - Suggestion 2
 - Suggestion 3
 - Suggestion 4

- REST API
- Proof-of-concept
 - Suggestion 1
 - Suggestion 2
 - Suggestion 4

B2. Persistence Layer

See file(s)

- Appendix_B2
 - <https://confluence.esss.lu.se/pages/viewpage.action?spaceKey=SW&title=New+Persistence+Layer+%28i.e.+database+schema%29+for+the+Naming+Service>
 - Created by Ricardo Fernandes, last modified on Jun 14, 2016
 - ns_database_schema.png