

Naming REST API – brief introduction

by lars.johansson@ess.eu, ICS Software

Table of Contents

Naming REST API – brief introduction.....	1
Introduction.....	1
Background.....	2
Concepts & Terminology.....	3
The purpose of Naming.....	3
What is a Name?.....	3
Visualization – names and structures.....	5
Lifecycle for names and structures.....	6
Vocabulary.....	9
REST API endpoints.....	12
Data – Access.....	12
Healthcheck.....	12
Report.....	12
Names.....	13
Path.....	13
Authorization.....	14
Structures.....	14
Path.....	14
Authorization.....	16
About searching.....	16
About data.....	17
Frequently Asked Questions (FAQ).....	18
Reference.....	19

Introduction

This document will give brief introduction to Naming REST API. First set of sections, such as Background and Concepts & Terminology, explains purpose of Naming and gives visualization of names & structures. Next set of sections attempts to give more practical information such as how to use Naming REST API, what data that is sent and received and what it means together with some questions and answers.

This document will mention persistence and storage but is not about it. Its focus is on REST API.

Background

It's been noticed that there is more and more focus on ability to service other applications, systems, services with information and act as a microservice. This means more focus on REST API. This includes having proper foundation for future with ability to support implementation of Naming convention.

Brief background on storage of data

- 1 entity for ESS name
- 1 entity per System structure level – System group, System, Subsystem (3 tables)
- 1 entity per Device structure level – Discipline, Device group, Device type (3 tables)
- *storage corresponds to data in real world. This helps understanding and maintainability of storage and application.*

Brief vocabulary

- names <---> ESS names
- structures <---> System structure, Device structure

Noteworthy

- REST API with Swagger UI
- No ordinary UI available
- Json format used for data that is sent and received
- Not about authentication/authorization. Intention is to have division of possibilities, e.g. visitor / user / administrator.

In various places are framed boxes with information that is noteworthy and may help understanding

Note!

Text that is noteworthy, summary, help & more

Concepts & Terminology

Concepts & Terminology are summarized below with *text in italic*.

The application is commonly referred to as *Naming*, *Naming tool*, *Naming convention tool*. It is a web application that includes a REST API for read purposes and it is connected to a storage of names and structures. Naming is the most common way to refer to both application and system as a whole.

The application is split into a backend part and a frontend part, known as *Naming backend* and *Naming frontend*.

The purpose of Naming

Handle Naming of ESS wide physical and logical devices according to ESS Naming Convention

What is a Name?

<i>ESS Name</i>	<i>System structure</i>	<i>Device structure</i>
	<i>Which part of the facility does the device provide service to?</i>	<i>What kind of service does the device provide?</i>
<i>Must refer to System structure</i>	<i>1 System Group</i>	<i>1 Discipline</i>
<i>May refer to Device structure</i>	<i>2 System</i>	<i>2 Device Group</i>
<i>May have index for instance</i>	<i>3 Subsystem</i>	<i>3 Device Type</i>

System structure and Device structure are hierarchies with 3 levels each. A 3rd level entry refers to 2nd level entry that refers to 1st level entry. A name must refer to System structure (arbitrary level) and may refer to Device structure (level 3). If Device structure is referred to, then index is to be set.

An entry, for name and structure, has a common set of attributes such as uuid, name, mnemonic, description, comment, when requested and by whom, when processed and by whom, etc.

Each entry is identified by its universal identifier, also known as uuid. It is the common denominator to keep track of an entry through its lifecycle, e.g. when the entry is created, updated, deleted. An entry usually has another attribute called *mnemonic* which is a short name that is to be unique in its namespace. This attribute is called *mnemonic* for an entry in System structure or Device structure. For a names entry, this attribute is called *index*.

In addition, there is an attribute called name equivalence or mnemonic equivalence. This is derived from name or mnemonic by taking similar-looking characters into account and helps to ensure that name and mnemonic is unique within its namespace.

Mnemonic equivalence

o, O, 0 considered same from equivalence point-of-view

i, I, l, L, 1 considered same from equivalence point-of-view

leading 0 numerical characters removed

Uniqueness of an entry

Each entry has a unique uuid that acts as key in its line of history. This applies for both name and structure entries.

An entry in names, System structure, Device structure must have attributes such that it is unique in its namespace at any given time. This is achieved with proper values for mnemonic and index.

Namespace for a name entry is all valid names.

Namespace for structure entry is its hierarchy.

Rules for names and structures

Structures

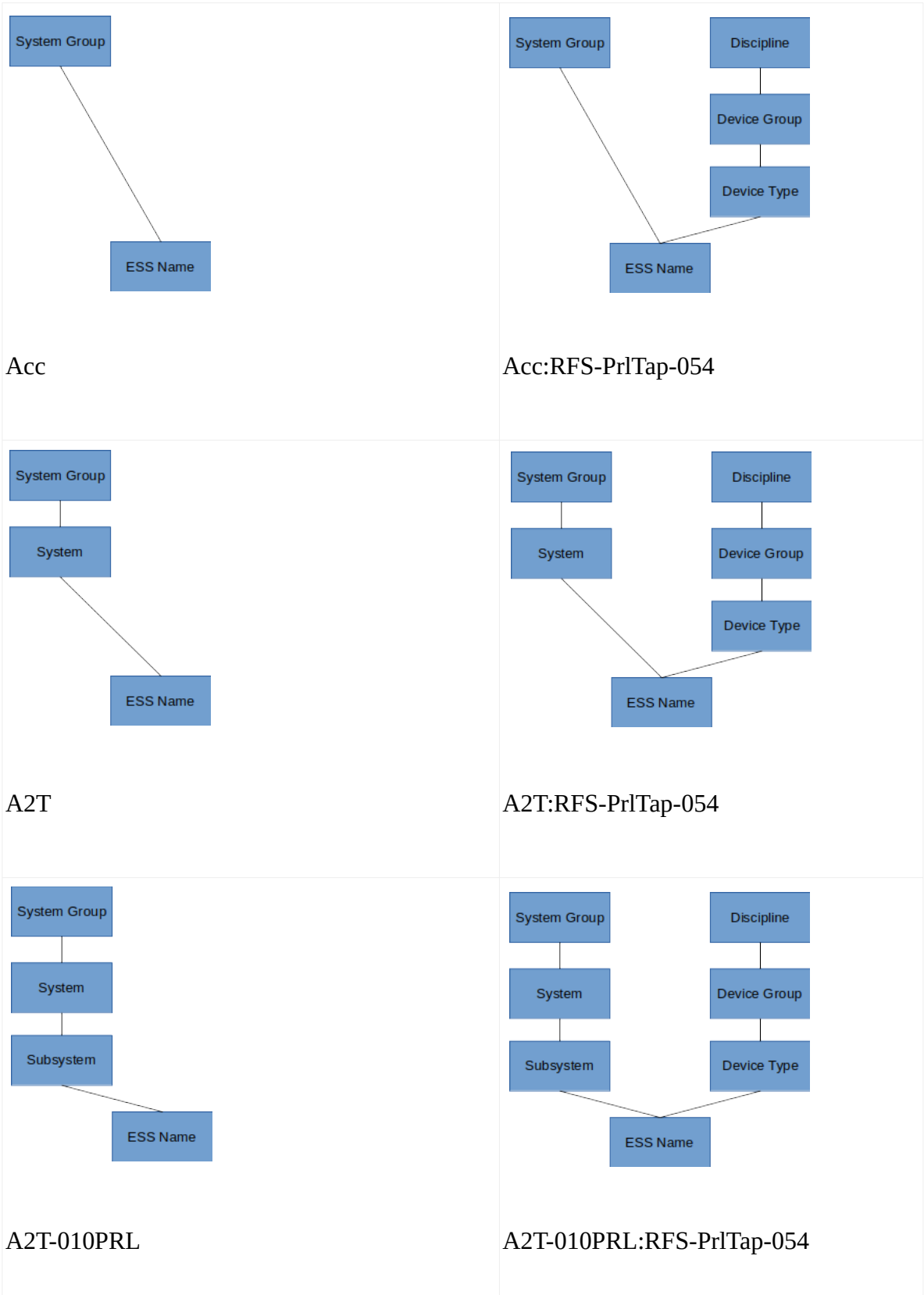
System, Subsystem, Discipline, Device Type must have mnemonic.

System Group may have mnemonic.

Device Group must not have mnemonic.

<i>ESS Name</i>	<i>=</i>	<i>System Group</i>				
		<i>System Group</i>	<i>+</i>	<i>Device Type</i>	<i>+</i>	<i>Index</i>
		<i>System</i>				
		<i>System</i>	<i>+</i>	<i>Device Type</i>	<i>+</i>	<i>Index</i>
		<i>Subsystem</i>				
		<i>Subsystem</i>	<i>+</i>	<i>Device Type</i>	<i>+</i>	<i>Index</i>

Visualization – names and structures



Lifecycle for names and structures

An entry for names is valid when it is created and saved. It does not go through an approval process. The entry may be modified multiple times. When entry is deleted, it has reached its end-of-life and may no longer be modified. A special case is if entry is considered legacy. This is the case if a parent is deleted. Then entry will remain but may no longer be modified except deleted. Any change may be done by user. The term legacy is introduced in the latest Naming convention.

An entry for System structure and Device structure needs to go through an approval process for any change. This includes create, modify, delete changes for an entry. The entry may be modified multiple times. When entry is deleted, it has reached its end-of-life and may no longer be modified. Any change is proposed by user and approved by administrator. A user may cancel request for change. An administrator may approve or reject request for change.

The whereabouts in the lifecycle for names and structures is handled with attributes status, latest and deleted. When an entry is deleted (names) or deleted and approved (structures), it has reached its end-of-line.

A change for an entry is handled such that a new entry is created. When the new entry is approved, attribute latest is set to true. Attribute latest for earlier entry is set to false. For any line of uuid, there may be zero or one entry with latest set to true.

Default behavior

Default behavior for Naming is to handle valid entries. Therefore old values are excluded unless history requested. This means that old values can not be browsed but must be requested. Latest entry in line of uuid is available for browsing.

Examples

Purpose of examples is to show lifecycle of names and structures. Therefore some columns are not shown, e.g. references to parents. A name has references to parents in System structure and Device structure. A structure entry has reference to a parent. Entries are grouped per uuid and shown in ascending order.

In addition, examples contain information from both user side and storage side.

Names

lifecycle	uuid	name	description	status	latest	deleted
obsolete	a	A2T-010PRL:RFS-PRLTap-052	comment	APPROVED	false	false
obsolete	a	A2T-010PRL:RFS-PRLTap-053	comment	APPROVED	false	false
active	a	A2T-010PRL:RFS-PRLTap-054	comment	APPROVED	true	false
obsolete	b	A2T-010PRL	comment 1	APPROVED	false	false
active	b	A2T-010PRL	comment 2	APPROVED	true	true
active	c	A2T	comment	APPROVED	true	false

Structures - e.g. System

lifecycle	uuid	mnemonic	description	status	latest	deleted
pending	m	A0T	comment	PENDING	false	false
obsolete	n	A1T	comment	PENDING	false	false
active	n	A1T	comment	APPROVED	true	false
obsolete	o	A2T	comment	PENDING	false	false
active	o	A2T	comment	APPROVED	true	false
obsolete	o	A3T	comment	PENDING	false	false
obsolete	o	A3T	comment	CANCELLED	false	false
obsolete	p	A4T	comment	PENDING	false	false
obsolete	p	A4T	comment	APPROVED	false	false
obsolete	p	A5T	comment	PENDING	false	false
obsolete	p	A5T	comment	REJECTED	false	false
obsolete	p	A5T	comment a	PENDING	false	false
active	p	A5T	comment a	APPROVED	true	false
obsolete	q	A6T	comment	PENDING	false	false
active	q	A6T	comment	APPROVED	true	false
obsolete	q	A6T	comment	PENDING	false	true
obsolete	q	A6T	comment	REJECTED	false	true
obsolete	r	A7T	comment	PENDING	false	false
obsolete	r	A7T	comment	APPROVED	false	false
obsolete	r	A7T	comment	PENDING	false	true
deleted	r	A7T	comment	APPROVED	true	true

Each time a modification is requested or processed, it will result in a new entry.

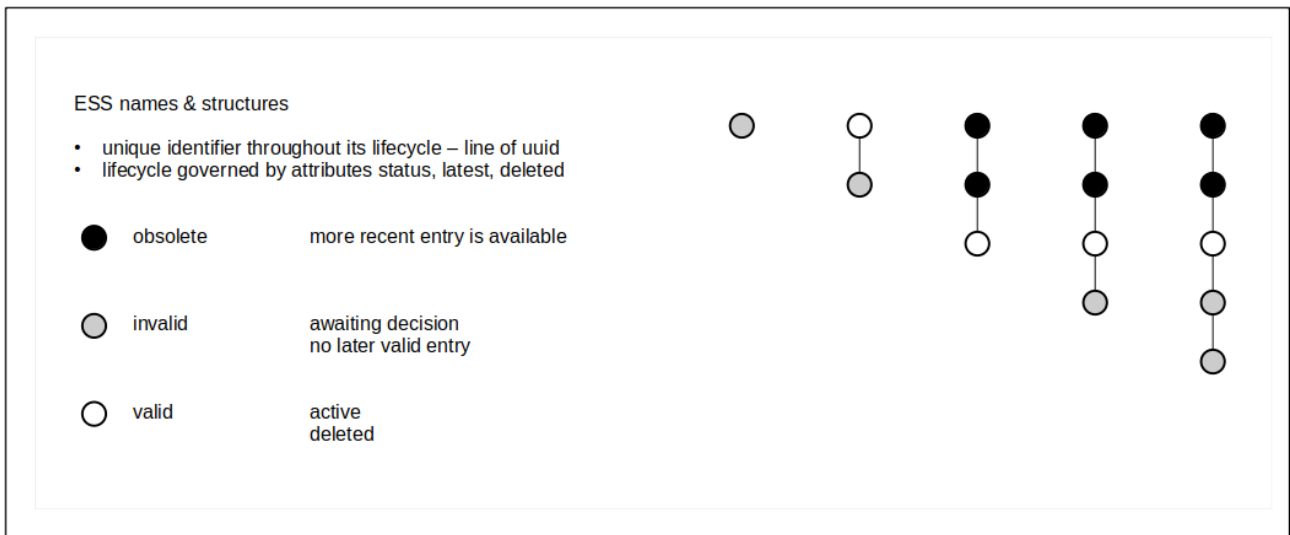
Each time a modification is processed (Structure), it will result in a new entry.

There is approval process for Structure but there is no approval process for Name.

When a modification is approved, its status attribute will be set to APPROVED and latest attribute set to true. For any given line of uuid, there may be 0 or 1 entry with latest attribute set to true. Latest attribute indicates if entry is latest and approved in its line of uuid.

An entry with attribute deleted set to true that has been APPROVED has reached its end-of-line.

Rows with gray background color are considered history and are excluded unless history is requested. Rows become history when there is a more recent entry with status APPROVED and latest set to true. History for an entry may be requested through its uuid.



The lifecycle for names and structures are handled with attributes status, latest, deleted as shown in examples above.

status – APPROVED, CANCELLED, REJECTED, PENDING

latest – true, false

deleted – true, false

The lifecycle may be simplified for easier handling, in particular for when information is read.

Combinations of status, latest, deleted may be handled by finite set of values

ACTIVE	status = APPROVED, latest = true, deleted = false
DELETED	status = APPROVED, latest = true, deleted = true
LEGACY	entry is ACTIVE but with deleted parent
OBSOLETE	entry is outdated (more recent entry available) or CANCELLED, REJECTED
PENDING	entry has modification that is requested but not yet processed

A value that no longer is valid corresponds to OBSOLETE.

Note that user needs to set values for attributes status, latest, deleted when an entry is created, modified or deleted.

Expected values for attributes status, latest, deleted when an entry is created, modified or deleted

create – status PENDING, latest false, deleted false

modify – status PENDING, latest false, deleted false

deleted – status PENDING, latest false, deleted true

Vocabulary

Vocabulary below with headlines with *text in italic*.

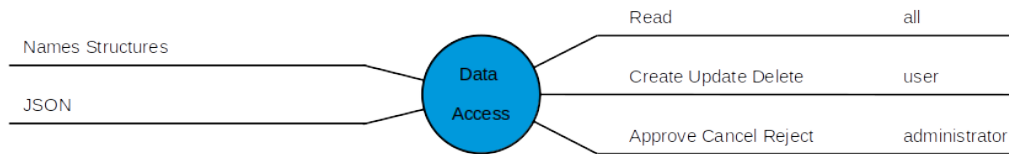
What	Description	Example
<i>Names & Structures</i>		
Name	Name for ESS physical and logical device	A2T-010PRL:RFS-PRLTap-054
System structure	Which part of the facility does the device provide service to?	A2T-010PRL
Device structure	What kind of service does the device provide?	Cryo-TT
System group	System structure level 1	Accelerator (Acc)
System	System structure level 2	Accelerator to Target (A2T)
Subsystem	System structure level 3	01 Phase Reference Line (010PRL)
Discipline	Device structure level 1	RF Systems (RFS)
Device group	Device structure level 2	Phase Reference Line
Device type	Device structure level 3	Phase Reference Line Tap (PRLTap)
<i>Concepts & Terminology</i>		
ESS Naming Convention	Rules for naming ESS Systems and Devices and Components in EPICS-based control system.	https://chess.esss.lu.se/enovia/link/ESS-0000757/21308.51166.45568.45993/valid
Equivalence	Derived from name or mnemonic by taking similar-looking characters into account and helps to ensure that name and mnemonic is unique within its namespace.	o, O, 0 - considered same from equivalence point-of-view i, I, l, L, 1 - considered same from equivalence point-of-view leading 0 numerical characters removed
Lifecycle of names and structures	The lifecycle of ESS name and structure entries. Each entry has a unique identifier throughout its lifecycle. The lifecycle is governed by attributes status, latest, deleted.	An entry that is deleted and approved may no longer be updated (or revived).
Line of uuid	A collection of ESS name or structure entries that share the same identifier and together make up an entry's history.	
Legacy name	An ESS name is considered legacy if it is active and has a deleted parent.	

Namespace	Line of uuid from top level to bottom level, for system structure and device structure, respectively. An index or a mnemonic or mnemonic equivalence may exist only once in a namespace for entries that are approved, latest, not deleted. Namespace for a name entry is all valid names. Namespace for a structure entry is its hierarchy.	
Rules for names and structures	<p>System, Subsystem, Discipline, Device Type must have mnemonic.</p> <p>System Group may have mnemonic.</p> <p>Device Group must not have mnemonic.</p>	
<i>REST API methods</i>		
POST	Http request method for create	create
GET	Http request method for read	read, find, search, equivalence, exists
PUT	Http request method for update	update
DELETE	Http request method for delete	delete
PATCH	Http request method for partial update	approve, cancel, request
<i>REST API schemas</i>		
Name element	A collection of fields that represent an ESS name entry (comprehensive). From server to client.	
Name command element	A collection of fields that represent an ESS name entry (minimum). From client to server. Purpose to simplify communication client to server.	
Structure element	A collection of fields that represent an ESS system structure or device structure entry (comprehensive). From server to client.	

Structure command element	A collection of fields that represent an ESS system structure or device structure entry (minimum). From client to server. Purpose to simplify communication client to server.	
<i>REST API fields (sub-selection)</i>		
Type	Kind of structure.	SYSTEMGROUP, SYSTEM, SUBSYSTEM, DISCIPLINE, DEVICEGROUP, DEVICETYPE
Index (Instance index)	A string. May be considered mnemonic for a name. To distinguish devices of the same type in the same system. Two different set of rules for index are identified for the <i>Scientific</i> and <i>P&ID</i> disciplines.	
Mnemonic	A set of characters and numbers to identify an entry in system structure and device structure.	
Mnemonic equivalence	A mnemonic with rules for equivalence applied.	
Status	Status for entry in hierarchy of names and structures	APPROVED, CANCELLED, REJECTED, PENDING
Latest	To show if entry is latest in its line of (uuid) entries.	true, false
Deleted	To show if entry is deleted in its line of (uuid) entries.	true, false
<i>REST API media type</i>		
application/json	Supported	
application/xml	Not supported	
<i>Authentication & authorization</i>		
Authentication	ESS username and password	
Authorization	None (read), User, Administrator	<p>None – Read-only access</p> <p>User – All operations for names. Propose create, update, delete for structure entries + cancel a proposal.</p> <p>Administrator – All operations for names and structures.</p>

REST API endpoints

Data – Access



Healthcheck

HTTP method	Path & Query string	Description
GET	/healthcheck	Perform healthcheck for Naming application in general and healthcheck endpoint in particular. To be used mainly for checking HTTP response code, in particular HTTP STATUS OK - 200.

Report

HTTP method	Path & Query string	Description
GET	/report/about	About Naming. Return report about Naming (text). Content <ul style="list-style-type: none">Metrics for Naming<ol style="list-style-type: none">OverviewESS namesSystem structureDevice structureDevice structure - P&ID Disciplines

Names

Path

/api/v1/names

HTTP method	Path & Query string	Description
POST		Create names by array of name element commands. Return array of name elements for created names.
POST	/upload	Create names by upload Excel file. Return Excel file with name elements for created names.
GET		Find valid names (search). Return paged array of name elements.
GET	/download	Find valid names (search). Return Excel file with paged list of name elements.
GET	/name	Find valid names by name or uuid (search). Return paged array of name elements.
GET	/systemStructure/{mnemonicpath}	Find valid names by system structure mnemonic path (search). Return paged array of name elements.
GET	/deviceStructure/{mnemonicpath}	Find valid names by device structure mnemonic path (search). Return paged array of name elements.
GET	/history	Find history for name by uuid (exact match). History consists of lines of uuid. The line of uuid is not broken in retrieving history. The line of uuid is not broken in retrieving history. If combination of parameters is found in name entries, the entire lines of uuid are returned. Return paged array of name elements.
GET	/history/{uuid}	Find history for name by uuid (exact match). History consists of line of uuid. The line of uuid is not broken in retrieving history. If the uuid is found in a name entry, the entire line of uuid is returned. Return paged array of name elements.
GET	/equivalence/{name}	Return name equivalence for name.
GET	/exists/{name}	Return if name exists (exact match). Response is true if name exists, false otherwise.
GET	/isLegacy/{name}	Return if name is legacy name (exact match). A name is considered legacy name if one or more of its parents is deleted.
GET	/isValidToCreate/{name}	Return if name is valid to create (exact match). Method answers question 'would it be ok to create given name?'.
PUT		Update names by array of name element commands. Return array of name elements for updated names.
PUT	/upload	Update names by upload Excel file. Return Excel file with name elements for updated names.

DELETE		Delete names by array of name element commands. Return array of name elements for deleted names.
DELETE	/upload	Delete names by upload Excel file. Return Excel file with name elements for deleted names.

Authorization

HTTP method	Authorization	Description
GET	Not required	Read
POST, PUT, DELETE	User	Create, Update, Delete

Structures

Path

/api/v1/structures

HTTP method	Path & Query string	Description
POST		Create (propose) structures by array of structure element commands. Return array of structure elements for created structures (proposals).
POST	/upload	Create (propose) structures by upload Excel file. Return Excel file with structure elements for created structures (proposals).
GET	/type	Find valid structures (search). Return paged array of structure elements.
GET	/type/download	Find valid structures (search). Return Excel file with paged list of structure elements.
GET	/children/{uuid}	Find valid children structures by type and parent uuid (exact match). Return paged array of structure elements.
GET	/mnemonic/{mnemonic}	Find valid structures by mnemonic (search). Return paged array of structure elements.
GET	/history	Find history for structure (search). History consists of lines of uuid. The line of uuid is not broken in retrieving history. If combination of parameters is found in structure entries, the entire lines of uuid are returned. Return paged array of structure elements.
GET	/history/{uuid}	Find history for structure by uuid (exact match). History consists of line of uuid. The line of uuid is not broken in retrieving history. If the uuid is found in a structure entry, the entire line of uuid is returned. Return paged array of structure elements.

GET	/equivalence/{mnemonic}	Return mnemonic equivalence for mnemonic.
GET	/exists/{type}/{mnemonicpath}	Return if mnemonic path exists in structure (exact match). Response is true if mnemonic path exists, false otherwise.
GET	/isValidToCreate/{type}/{mnemonicpath}	Return if name is valid to create (exact match). Method answers question 'would it be ok to create given name?'.
PUT		Update (propose) structures by array of structure element commands. Return array of structure elements for updated structures (proposals).
PUT	/upload	Update (propose) structures by upload Excel file. Return Excel file with structure elements for updated structures (proposals).
DELETE		Delete (propose) structures by array of structure element commands. Return array of structure elements for deleted structures (proposals).
DELETE	/upload	Delete (propose) structures by upload Excel file. Return Excel file with structure elements for deleted structures (proposals).
PATCH	/approve	Approve structures (proposals) by array of structure element commands. Return array of structure elements for approved structures. Name is automatically created name when creation of system structure is approved.
PATCH	/approve/upload	Approve structures (proposals) by upload Excel file. Return Excel file with structure elements for approved structures. Name is automatically created name when creation of system structure is approved.
PATCH	/cancel	Cancel structures (proposals) by array of structure element commands. Return array of structure elements for cancelled structures.
PATCH	/cancel/upload	Cancel structures (proposals) by upload Excel file. Return Excel file with structure elements for cancelled structures.
PATCH	/reject	Reject structures (proposals) by array of structure element commands. Return array of structure elements for rejected structures.
PATCH	/reject/upload	Reject structures (proposals) by upload Excel file. Return Excel file with structure elements for rejected structures.

Authorization

HTTP method	Authorization	Description
GET	Not required	Read
POST, PUT, DELETE	User	Create, Update, Delete
PATCH	Administrator	Approve, Cancel, Reject

About searching

Exact match

- Is exact match = no search

Search

- Default behavior is exact match
- No regex
- Two additional characters may be used to help search and may be written anywhere in search string to give regex-like behavior
 - `_` underscore, 0 or 1 occurrences of any character
 - `%` percent, any number of any character
 - e.g.
 - A2T-010PRL:RFS-PRLTap-054
 - A2T-010PRL:RFS-PRLTap-0_ will not give match
 - A2T-010PRL:RFS-PRLTap-0__ will give match
 - A2T-010PRL:RFS-PRL% will give match

About data

What is sent to and received from Naming REST API

- a string
- json

Examples

Name and NameElement

A2T-010PRL:RFS-PRLTap-054

System structure

- Accelerator
- Accelerator to Target
- 01 Phase Reference Line

Device structure

- RF Systems
- Phase Reference Line
- Phase Reference Line Tap

Index

- 054

json

- {"uuid":"07bce0ae-0947-47c8-941e-cc76678fd29a","description":null,"status":"APPROVED","latest":true,"deleted":false,"when":"2017-10-20T12:53:27.229+00:00","who":"johannorin","comment":null,"systemgroup":null,"system":null,"subsystem":"c2fce615-ed5d-40f9-8fb5-0b91502536e5","devicetype":"bb1e68a6-e233-4595-ae88-f9186b6760c6","systemstructure":"A2T-010PRL","devicestructure":"RFS-PRLTap","index":"054","name":"A2T-010PRL:RFS-PRLTap-054"}

Rules

- A name must have exactly one system structure parent, either systemgroup or system or subsystem
- A name may have device structure parent, devicetype

In name above

- subsystem uuid refers to subsystem *01 Phase Reference Line* that in turn refers to system *Accelerator to Target* that in turn refers to system group *Accelerator*

- device type uuid refers to device type *Phase Reference Line Tap* that in turn refers to device group *Phase Reference Line* that in turn refers to discipline *RF Systems*

Since name above was created, implementation of Naming was changed so that description and comment are mandatory.

Structure and StructureElement

Accelerator to Target A2T

json

- {"uuid":"e67a497c-9c55-4942-97fc-700c8ec56031","description":"The Accelerator to Target Station interface including the dogleg","status":"APPROVED","latest":true,"deleted":false,"when":"2016-07-04T10:06:38.873+00:00","who":"danielpisofernandez","comment":"Approved by Daniel Piso","type":"SYSTEM","parent":"4262e1e7-2444-412e-83d7-aeabf58262c6","name":"Accelerator to Target","mnemonic":"A2T","mnemonicpath":"Acc-A2T","level":2}

Frequently Asked Questions (FAQ)

Topics / questions / answers, no particular order

- Capabilities of REST API
 - All operations in Naming are available in REST API
- Json format used for data that is sent and received
 - Data is provided or received as json, as single entry of arrays of entries,
- Retrieval available with exact match and search. Default for search is exact match but may be adjusted for true search. Regex is not available. There are instead two special characters to use for search and regex-like behavior.
 - _ underscore, 0 or 1 occurrences of any character
 - % percent, any number of any character
- Retrieval available with abilities
 - search on individual fields
 - sorting
 - pagination
- Index for a name
 - Field is alphanumerical. Existing names usually have index. Values include alphabetical, numerical, alphanumerical. When index is to be set, it is to be set explicitly and not auto-generated.

- Legacy name
 - A legacy name is a name for which one or both parents (System structure, Device structure) have been deleted. In current Naming, a name is deleted when a parent of arbitrary level is deleted. This is not the intention of the latest Naming convention. Instead the name will keep on living when one or both of its parents are deleted but the only change that is possible is deletion. However the name will keep on living until it's explicitly deleted.
- History
 - History consists of lines of uuid. The line of uuid is not broken in retrieving history. If combination of parameters is found in name or structure entries, the entire lines of uuid are returned.
- Validation
 - Ability to validate modifying operation before invoking modifying operation. Modifying operation internally use same validation.

Reference

Naming convention

- <https://chess.esss.lu.se/enovia/link/ESS-0000757/21308.51166.45568.45993/valid>