Shift Handover

There's a project to develop a new and improved application for handling shift handover between day and night shifts offshore. The main objective is to overhaul the current solution (eLog/eLogbook) with focus on new design, enhanced user friendliness and integrations with other applications/systems. The main parts of the application are a shift log and the handover itself, where the shift log will be a log of events that have happened during the shift and the handover will include events that are deemed relevant for the next shift to have knowledge about. There's an ambition to add auto-generation of relevant events in the log based on inputs/events from the control systems, which should be found in CDF once the Digital Foundation is up an running. This notebook is made to map relevant events that are accessible through CDF.

Config/libraries:

In [1]:

```
import math, csv, re, datetime
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from getpass import getpass                # Allows for an entry field to avoid hard-coded API keys
from cognite.client import CogniteClient   # Python SDK
from cognite.client.data_classes.assets import AssetList
from cognite.client.data_classes.events import EventList
from cognite.client.utils import timestamp_to_ms, ms_to_datetime

%matplotlib widget
```

Setup of Cognite SDK for communication with CDF (will ask for API-key):

In [2]:

```
api_key = getpass()
client = CogniteClient(
    api_key = api_key,
    project="akerbp",
    client_name="DSHub",
    base_url="https://api.cognitedata.com"
)
```

Statics:

In [3]:

```
class Alarm:
    def __init__(self, alarm_id, new_state = None, alarm_state = None):
        self.id = alarm_id
        self.new_state = new_state
        self.alarm_state = alarm_state


class Component:
    def __init__(self, name, area, eq_group, desc, alt_name=None):
        self.name = name
        self.area = area
        self.equipment_group = eq_group
        self.description = desc
        self.assets = None
        self.exact_match = False
        self.alt_name = alt_name


class HandoverAsset:
    def __init__(self, asset, exact_match):
        self.name = asset.name
        self.parent = asset
        self.events = EventList([])
        # If asset has a state subtag (e.g. a running or open status: ..._31 or ...-31)
        self.state_subtag = asset.name[-2:] if re.search(f'[-_][0-9][0-9]$', asset.name) else None
        self.exact_match = exact_match


class Field:
    def __init__(self, name, data_set_id=None):
        self.name = name
        self.data_set_id = data_set_id
        self.assets = None
        self.main_components = None

    def abbr(self):
        return {
            'Alvheim': 'ALV',
            'Ivar Aasen': 'IAA',
```

```
            'Skarv': 'SKA',
            'Ula': 'ULA',
            'Valhall': 'VAL'
        }[self.name]


class Project:
    def __init__(self, name):
        self.name = name
        self.fields = {}
        self.main_components = {}

    def append_field(self, name):
        self.fields.update({{
                'Alvheim': 'ALV',
                'Ivar Aasen': 'IAA',
                'Skarv': 'SKA',
                'Ula': 'ULA',
                'Valhall': 'VAL'
            }[name]: Field(name)
        })

    def data_sets(self):
        sets = []
        for field in self.fields.values():
            if field.data_set_id:
                sets.append(field.data_set_id)
        return sets


handover = Project("Shift Handover")
handover.append_field('Alvheim')
handover.append_field('Ivar Aasen')
handover.append_field('Skarv')
handover.append_field('Ula')
handover.append_field('Valhall')
```

Find the SAP Functional Location for the source of an event

In [4]:

```
# This function retrieves a CDF event by its event id
def get_event(id):
    return client.events.retrieve(id=id)

# This function retrieves a CDF asset by its asset id
def get_asset(id):
    return client.assets.retrieve(id=id)
```

In [5]:

```
# This function extracts tag names from a text string
# Inputs:
# string - text string (e.g. '23-KA-9103-M01_CB_travel_alarm')
# field - all caps abbreviation of an Aker BP asset (e.g. 'VAL')
#
# Output:
# list of found tag names in string, matching the given fields tagging convention
# (e.g. ['23-KA-9103'])

def extract_tag_names(string, field):
    def default(string, field):
        print(f"There is no function for extracting tags for {field}")
        return None

    # Ula tags usually follow this format: A(AAA)-NNNN(N)(A)-(A)
    def extract_ula_tag(string, field):
        return re.search(r'[A-Z]+-\d+([A-Z][A-Z]?|(-[A-Z]\b)?)', string)

    # Valhall tags usually follow this format: NN-AA(A)-NNNN(NN)(A)
    def extract_valhall_tag(string, field):
        return re.search('[0-9][0-9]-[A-Z]+-\d+[A-Z]?[A-Z]?', string)

    switcher = {
        'ULA': extract_ula_tag,
        'VAL': extract_valhall_tag
    }

    x = re.split('_', string)
    matches = []
    for word in x:
        y = switcher.get(field, default)(word, field)
        if y:
            matches.append(y.group())
    if not matches: print(f'Could not identify any tags with {field} tagging convention in "{string}"')

    return matches
```

In [41]:

```
# NOTE: Running this cell will not work before main_components is built from a cell below
for comp in handover.main_components['VAL'][:10]:
    for asset in comp.assets[:1]:
        res = extract_tag_names(asset.name, 'VAL')
        if res: print(f'{res} <- {asset.name}')
```

```
['11-PA-9102'] <- 11-PA-9102-M01_Motor_startup_supervision_MOTstart
['11-PA-9138'] <- 11-PA-9138-M01-36
['11-PA-9101A'] <- 11-PA-9101A-M01-24
['11-PA-9101B'] <- 11-PA-9101B-M01-24
['18-PA-8010'] <- 18-PA-8010-M01-13
['18-PA-8030'] <- 18-PA-8030-M01_31__
['18-PA-8050'] <- 18-PA-8050-M01_31__
['18-PA-8070'] <- 18-PA-8070-M01-04_E
['84-KF-8030'] <- 84-KF-8030-M01_TRT
['23-KA-9101'] <- 23-KA-9101-M01_CB_multiple_plug_out
```

In [7]:

```
# This function looks for the field (Aker BP asset) of a CDF asset
def get_field_from_asset(asset):
    # The root asset usually holds information about the Aker BP installation
    root_asset = get_asset(asset.root_id)
    installation = root_asset.name

    if installation in ['VAL', 'VFN', 'VFS', 'VFW', 'HOD', 'HOP', 'VLA']:
        return 'VAL'
    elif installation in ['ULA', 'TAM']:
        return 'ULA'
    elif installation in ['SKA', 'ALV', 'IAA']:
        return installation

    print(f'Found unexpected field from root asset: {installation}')
    return installation
```

In [8]:

```
# This function recursively searches for a parent asset of an event that is not an asset created by e.g. signal tags/subtags/softt
ags/etc.
#
# Example hierarchy:
# Z-0501 (id: 8289785573229588)          <-- True parent
#  LP-0561 (id: 4103300555352816)        <-- Some other ancestor (tag does not match)
#    Z-0501-01 (id: 6791812333680596)    <-- Grandparent that comes from Aveva (still has signal tag)
#      Z-0501-01 (id: 4728001758341600)  <-- Parent asset in OPC UA dataset (has signal tag)
#        (event) Abnormal condition (source: Z-0501-01, id: 7491014714576560)

def get_true_parent(event, print_hierarchy = False):
    parent = None
    source_name = event.metadata["source"]

    # Store order of hierarchy for printing and troubleshooting purposes
    hierarchy = [[f'(event) {event.description} (source: {source_name}, id: {event.id})'] for _ in range(len(event.asset_ids))]

    # Loop through all related CDF assets for the given event
    for i, asset_id in enumerate(event.asset_ids):
        temp_asset = get_asset(asset_id)

        # List of potential tag names following the given fields tagging convention, usually just one
        tag_names = extract_tag_names(source_name, get_field_from_asset(temp_asset))

        # Recursively search for a parent asset that matches with the desired tag name (up the CDF asset hierarchy)
        j = 0
        while not parent and j < 5:
            hierarchy[i].append(f'{temp_asset.name} (id: {temp_asset.id})')    # For printing and troubleshooting purposes
            for name in tag_names:
                # If the parent assets name is part of or equal to the desired tag name, we have found a match
                # (e.g. if the found asset has a main equipment tag name, but the desired tag name is a redundancy tag with A/B/et
c. at the end, we still consider it a match)
                if temp_asset.name in name:
                    parent = temp_asset
                    if name != temp_asset.name: print(f'{temp_asset.name} will be used instead of {name}')   # To verify that non-
exact matches still are relatives

                    # For printing and troubleshooting purposes
                    if print_hierarchy:
```

```
                    for k in range(len(hierarchy[i])):
                        indent = k*"  "
                        print(indent + hierarchy[i][-(k+1)])

                return parent
            temp_asset = get_asset(temp_asset.parent_id)
            j += 1

        if not parent: print(f'Could not find true parent of event {event.id} after {j} recursive steps using asset {asset_id} ({i
+1}/{len(event.asset_ids)})')
    print(f'No true parents were found for event {event.id} (event had {len(event.asset_ids)} related CDF asset(s))')

    #if print_hierarchy: ...

    return parent
```

In [9]:

```
# This function recursively searches for a parent asset that holds information that can determine a SAP FLOC prefix
def get_floc_prefix(asset):
    temp_asset = asset

    # FLOC prefix can often be found from an assets 'PLATFORM CODE' attribute
    prefix = temp_asset.metadata.get('PLATFORM CODE', '')

    # Recursively search for a parent asset holding the 'PLATFORM CODE' property (up the CDF asset hierarchy)
    i = 0
    while not prefix and i < 5:
        temp_asset = get_asset(temp_asset.parent_id)
        prefix = temp_asset.metadata.get('PLATFORM CODE', '')
        i += 1

    if not prefix: print(f'Could not find FLOC prefix for asset {asset.id} after {i} recursive steps')
    return prefix
```

In [10]:

```
# This function returns the SAP FLOC of a CDF event
def get_floc(event, print_hierarchy = False):
    temp_floc = event.metadata["source"]
    true_parent = get_true_parent(event, print_hierarchy)
    if true_parent:
        temp_floc = true_parent.name
        prefix = get_floc_prefix(true_parent)
        if prefix:
            temp_floc = prefix + "-" + temp_floc
    return temp_floc
```

In [11]:

```
test_event = get_event(7491014714576560)
test_event = get_event(8765205970484357)
get_floc(test_event, True)
```

```
23-KA-9103 (id: 8571004476315909)
  23-KA-9103-M01_CB_travel_alarm (id: 361499351463819)
    (event) Acknowledge of CFN condition on object 23-KA-9103-M01_CB_travel_alarm has been requested. (source: 23-KA-
9103-M01_CB_travel_alarm, id: 8765205970484357)
```

Out[11]:

```
'VPH-23-KA-9103'
```

In [12]:

```
def print_event(event):
    time = event.metadata.get('activeTime', event.metadata['time'])
    floc = get_floc(event)
    object_desc = event.metadata.get('ObjectDescription', '**no object description**')
    description = f'{event.metadata["message"]}, {event.metadata.get("conditionName", "")}'
    severity = event.metadata['severity']
    print(f'    {time} | {floc} | {object_desc} | {description} | {severity} (id: {event.id})')
```

In [13]:

```
print_event(test_event)
```

```
    6/15/2022 7:38:08 AM | VPH-23-KA-9103 | PH MV-COMP.M. FEEDER/ CB_travel_alarm | Acknowledge of CFN condition on o
bject 23-KA-9103-M01_CB_travel_alarm has been requested.,  | 200 (id: 8765205970484357)
```

Extract main components:

In [14]:

```
csv_files = {
    "ALV": 'main_components_alvheim.csv',
    "IAA": 'main_components_ivar_aasen.csv',
    "SKA": 'main_components_skarv.csv',
    "ULA": 'main_components_ula_abb_tags.csv',
    "VAL": 'main_components_valhall.csv'
}

for field in csv_files:
    with open(f'data/{csv_files[field]}', 'rt', encoding='utf-8-sig') as f:
        data = csv.reader(f, delimiter=";")
        header = next(data)
        handover.fields[field].main_components = []
        for row in data:
            # If .csv-file has alternative tags (e.g. ABB tags vs Aveva tags like Ula)
            alt_name = None
            if len(row) > 4:
                # If there's only one alternative ABB tag, make it an alternative name
                if not row[5]:
                    if row[4]: alt_name = str(row[4]).strip()
                # If there's multiple ABB tags (or subtags to main equipment), add them as main components with 'instrument' postf
ix
                else:
                    for name in [name for name in row[4:] if name]: handover.fields[field].main_components.append(Component(str(na
me).strip(), str(row[0]), str(row[1]), str(row[3]).strip()+' instrument'))
                    continue
            handover.fields[field].main_components.append(Component(str(row[2]).strip(), str(row[0]), str(row[1]), str(row[3]).str
ip(), alt_name))
        print(f'{handover.fields[field].name}: {len(handover.fields[field].main_components)} main component tags')
```

```
Alvheim: 233 main component tags
Ivar Aasen: 71 main component tags
Skarv: 517 main component tags
Ula: 56 main component tags
Valhall: 54 main component tags
```

In [15]:

```
#data_sets = client.data_sets.list(limit=-1)
#data_sets[30:60]
```

Digital Foundation OPC UA dataset IDs:

In [16]:

```
handover.fields['ULA'].data_set_id = 2086908079872503
handover.fields['VAL'].data_set_id = 140572846698809

#1525574569706251 #ABB Ability - Alarms and Events Valhall
#3874712032478167 #ABB Ability - Alarms and Events ULA
```

Pull all assets within the OPC UA / Digital Foundation datasets:

In [17]:

```
for field in handover.fields.values():
    if field.data_set_id:
        field.assets = client.assets.list(data_set_ids=[field.data_set_id], limit=-1)
        print(f'{field.name} has {len(field.assets)} total assets in CDFs Digital Foundation dataset')
```

```
Ula has 18930 total assets in CDFs Digital Foundation dataset
Valhall has 40959 total assets in CDFs Digital Foundation dataset
```

In [18]:

```
#handover.fields['ULA'].assets[1]
```

Search through all assets for relevant tags and retrieve only the assets related to the main component lists:

In [19]:

```
for field in handover.fields.values():
    if field.assets:
        handover.main_components[field.abbr()] = []
        count = 0
        exact_matches = 0
        pre_count = 0
        post_count = 0
        for comp in field.main_components:
            found = False
            for asset in field.assets:
                exact_match = False
                for word in re.split("_", asset.name):
                    name_match = re.search(f'^{comp.name}$', word) or re.search(f'^{comp.name}[-a-zA-Z]', word)
                    alt_name_match = re.search(f'^{comp.alt_name}$', word) or re.search(f'^{comp.alt_name}[-a-zA-Z]', word) if com
p.alt_name else False
                    if name_match or alt_name_match:
                        if comp.name == asset.name or comp.alt_name == asset.name:
                            exact_match = True
                            if not comp.exact_match:
                                comp.exact_match = True
                                exact_matches += 1
                            else:
                                print(f'Multiple CDF assets were found exactly matching main component {comp.name} (alt_name: {com
p.alt_name})')
                        if not found:
                            comp.assets = []
                            handover.main_components[field.abbr()].append(comp)
                            found = True
                            if name_match: pre_count += 1
                        handover.main_components[field.abbr()][-1].assets.append(HandoverAsset(asset, exact_match))
                        count += 1
                        break
        print(f'\n{len(handover.main_components[field.abbr()])} of {len(field.main_components)} main component tags found in {fiel
d.name} asset list ({pre_count} without ABB tags)')
        print(f'These tags matched with {count} assets ({exact_matches} exact matches)')
```

Multiple CDF assets were found exactly matching main component P-1101-A (alt_name: PM-1101A)
Multiple CDF assets were found exactly matching main component P-1101-B (alt_name: PM-1101B)

41 of 56 main component tags found in Ula asset list (8 without ABB tags)
These tags matched with 234 assets (35 exact matches)

35 of 54 main component tags found in Valhall asset list (35 without ABB tags)
These tags matched with 807 assets (0 exact matches)

In [20]:

```
tag_states = {1: 'START/STOP OR OPEN/CLOSE', 2: 'START OR OPEN (ON)', 3: 'STOP OR CLOSE (OFF)', 4: 'ESD/TRIP/SHUTDOWN', 5: 'PULSED START OR OPEN (ON)', 6: 'PULSED STOP OR CLOSE (OFF)', 7: 'INHIBIT/DISABLE START OR OPEN',
              8: 'INHIBIT/DISABLE STOP OR CLOSE', 9: 'CONTROLLER SETPOINT', 10: 'DIRECTION', 11: 'RESET TRIP/FAULT', 12: 'THYRISTOR CONTROL', 13: 'LOAD SHEDDING', 14: 'F&G TRIP/SHUTDOWN', 15: 'SPARE', 16: 'SPARE',
              17: 'SPARE', 18: 'SPARE', 19: '', 20: 'GEN. COMMAND SIGNAL', 21: 'GEN. COMMAND SIGNAL', 22: 'GEN. COMMAND SIGNAL', 23: 'GEN. COMMAND SIGNAL', 24: 'PSD TRIP SHUTDOWN', 25: 'GEN. COMMAND SIGNAL',
              26: 'GEN. COMMAND SIGNAL', 27: 'GEN. COMMAND SIGNAL', 28: 'GEN. COMMAND SIGNAL', 29: 'GEN. COMMAND SIGNAL', 30: 'GEN. COMMAND SIGNAL', 31: 'RUNNING OR OPEN', 32: 'STOPPED OR CLOSED',
              33: 'AVAILABLE OR READY', 34: 'COMMON ALARM/FAULT', 35: 'LOCAL/REMOTE', 36: 'TRIPPED (INTERNAL FAULT)', 37: 'TEST POSITION', 38: 'SERVICE POSITION', 39: 'RUNNING/STOPPED, OPEN/CLOSED (1=RUNNING,OPEN)',
              40: 'FAULT (INTERNAL PROTECTION)', 41: 'FAULT (INTERNAL PROTECTION)', 42: 'FAULT (INTERNAL PROTECTION)', 43: 'FAULT (INTERNAL PROTECTION)', 44: 'FAULT (INTERNAL PROTECTION)',
              45: 'FAULT (INTERNAL PROTECTION)', 46: 'ALARM OVERCURRENT', 47: 'ALARM SHORTCIRCUIT', 48: 'TEMP HIGH ALARM', 49: 'TEMP HIGH HIGH ALARM', 50: 'BUSBAR EARTHED', 51: 'ARC TRIP ON BUSBAR',
              52: 'LOSS OF MAIN POWER', 53: 'EARTH SWITCH OPEN/CLOSED', 54: 'CIRCUIT BEAKER OPEN/CLOSED', 55: 'GEN. STATUS SIGNAL', 56: 'GEN. STATUS SIGNAL', 57: 'GEN. STATUS SIGNAL', 58: 'GEN. STATUS SIGNAL',
              59: 'GEN. STATUS SIGNAL', 60: 'CURRENT (A)', 61: 'VOLTAGE (V)', 62: 'ACTIVE POWER (kW)', 63: 'REACTIVE POWER (kW)', 64: 'TEMPERATURE (degC)', 65: 'SPEED/FREQUENCY/TEMP',
              66: 'OPERATING/RUNNING TIME (days or hours)', 67: 'ENERGY CONSUMPTION (kWh or MWh)', 68: 'TORQUE/NOMINAL TORQUE (%)', 69: 'SPARE', 70: 'SPARE', 71: 'SPARE', 72: 'SPARE', 73: 'SPARE', 74: 'SPARE',
              75: 'SPARE', 76: 'SPARE', 77: 'SPARE', 78: 'SPARE', 79: 'SPARE', 80: 'GEN. MEASURING SIGNAL', 81: 'GEN. MEASURING SIGNAL', 82: 'GEN. MEASURING SIGNAL', 83: 'GEN. MEASURING SIGNAL',
              84: 'GEN. MEASURING SIGNAL', 85: 'GEN. MEASURING SIGNAL', 86: 'GEN. MEASURING SIGNAL', 87: 'GEN. MEASURING SIGNAL', 88: 'GEN. MEASURING SIGNAL', 89: 'GEN. MEASURING SIGNAL', 90: 'SPARE', 91: 'SPARE',
              92: 'SPARE', 93: 'SPARE', 94: 'SPARE', 95: 'SPARE', 96: 'SPARE', 97: 'SPARE', 98: 'SPARE', 99: 'SPARE'}
for field, comps in handover.main_components.items():
    n_subtag_assets = 0
    n_subtag_components = 0
    found_subtags = {}
    for comp in comps:
        subtag_found = False
        for asset in comp.assets:
            if asset.state_subtag:
                n_subtag_assets += 1
                subtag_found = True
                if int(asset.state_subtag) not in found_subtags:
                    found_subtags[int(asset.state_subtag)] = 1
                else:
                    found_subtags[int(asset.state_subtag)] += 1
```

```
        if subtag_found: n_subtag_components += 1
    print(f'\n{n_subtag_components} of {len(comps)} components had assets with state subtags at {field} ({n_subtag_assets} asset
s):')
    for state in range(1,100):
        if state in found_subtags:
            print(f'{"0"+str(state) if state < 10 else state} - {tag_states[state]}: {found_subtags[state]} assets')
```

41 of 41 components had assets with state subtags at ULA (116 assets):
01 - START/STOP OR OPEN/CLOSE: 17 assets
02 - START OR OPEN (ON): 33 assets
06 - PULSED STOP OR CLOSE (OFF): 8 assets
08 - INHIBIT/DISABLE STOP OR CLOSE: 34 assets
09 - CONTROLLER SETPOINT: 11 assets
10 - DIRECTION: 4 assets
13 - LOAD SHEDDING: 1 assets
18 - SPARE: 1 assets
27 - GEN. COMMAND SIGNAL: 2 assets
33 - AVAILABLE OR READY: 1 assets
34 - COMMON ALARM/FAULT: 1 assets
35 - LOCAL/REMOTE: 3 assets

33 of 35 components had assets with state subtags at VAL (182 assets):
01 - START/STOP OR OPEN/CLOSE: 5 assets
04 - ESD/TRIP/SHUTDOWN: 14 assets
05 - PULSED START OR OPEN (ON): 10 assets
06 - PULSED STOP OR CLOSE (OFF): 8 assets
09 - CONTROLLER SETPOINT: 2 assets
12 - THYRISTOR CONTROL: 1 assets
13 - LOAD SHEDDING: 14 assets
14 - F&G TRIP/SHUTDOWN: 2 assets
15 - SPARE: 2 assets
20 - GEN. COMMAND SIGNAL: 2 assets
24 - PSD TRIP SHUTDOWN: 18 assets
25 - GEN. COMMAND SIGNAL: 7 assets
31 - RUNNING OR OPEN: 2 assets
33 - AVAILABLE OR READY: 11 assets
34 - COMMON ALARM/FAULT: 1 assets
35 - LOCAL/REMOTE: 2 assets
36 - TRIPPED (INTERNAL FAULT): 14 assets
37 - TEST POSITION: 10 assets
38 - SERVICE POSITION: 10 assets
39 - RUNNING/STOPPED, OPEN/CLOSED (1=RUNNING,OPEN): 18 assets
48 - TEMP HIGH ALARM: 6 assets
49 - TEMP HIGH HIGH ALARM: 3 assets
53 - EARTH SWITCH OPEN/CLOSED: 4 assets
54 - CIRCUIT BEAKER OPEN/CLOSED: 4 assets
60 - CURRENT (A): 6 assets

62 - ACTIVE POWER (kW): 4 assets
65 - SPEED/FREQUENCY/TEMP: 2 assets

In [21]:

```
print(get_event(2592908100484675))
print(get_asset(812863580834318))
print(get_asset(482220262989013))
print(get_asset(1202379980374596))
```

```
{
    "external_id": "opcua_val_BBMAcdSgOECs1Q8C9KoJfg==",
    "data_set_id": 140572846698809,
    "start_time": "2022-01-27 10:08:04",
    "end_time": "2022-01-27 10:08:04",
    "type": "Simple",
    "description": "Normal",
    "metadata": {
        "severity": "409",
        "Emitter": "opcua_val_eo:s=e102c1d1-fcbf-4903-a4da-38738aada0cc",
        "eventCategory": "1258335080",
        "contextConfidenceScore": "1",
        "SourceName": "47-PS-12001-M01-01_",
        "Severity": "409",
        "source": "47-PS-12001-M01-01_",
        "eventType": "Simple",
        "contextAlgorithm": "Substring match - Other location",
        "type": "opcae",
        "message": "Normal",
        "version": "2.0.0",
        "EventData": "[{\"version\":\"2.0.0\"},{\"type\":\"opcae\"},{\"eventCounter\":55402946},{\"id\":\"71001304-a0
d4-4038-acd5-0f02f4aa097e\"},{\"source\":\"47-PS-12001-M01-01_\"},{\"time\":\"1/27/2022 10:08:04 AM\"},{\"eventType
\":\"Simple\"},{\"eventCategory\":1258335080},{\"eventCategoryName\":\"SimpleProcess(MB300)\"},{\"severity\":409},
{\"message\":\"Normal\"},{\"Class\":47},{\"ProcessSection\":4},{\"ObjectDescription\":\"FN SeawaterLiftPump\"}]",
        "eventCategoryName": "SimpleProcess(MB300)",
        "ProcessSection": "4",
        "contextAgent": "OPC UA contextualization pipeline",
        "SourceNode": "opcua_val_eo:s=e102c1d1-fcbf-4903-a4da-38738aada0cc",
        "eventCounter": "55402946",
        "Class": "47",
        "id": "71001304-a0d4-4038-acd5-0f02f4aa097e",
        "time": "1/27/2022 10:08:04 AM",
        "ObjectDescription": "FN SeawaterLiftPump"
    },
    "asset_ids": [
        812863580834318
    ],
    "id": 2592908100484675,
    "last_updated_time": "2022-01-27 10:42:20",
    "created_time": "2022-01-27 10:08:32"
}
```

```
{
    "external_id": "opcua_val_eo:s=b1f559e1-c1b8-40e4-93a1-f750c17a9b72",
    "name": "47-PS-12001-M01-01",
    "parent_id": 482220262989013,
    "parent_external_id": "VFN_47-PS-12001-M01-01",
    "description": "",
    "data_set_id": 140572846698809,
    "metadata": {
        "Description": "",
        "Id": "b1f559e1-c1b8-40e4-93a1-f750c17a9b72",
        "Name": "47-PS-12001-M01-01",
        "PNE.DataType": "boolean",
        "PNE.Value": "True",
        "TypeDefinition": "EdgeObjectType",
        "TypeId": "DO:OB",
        "Version": "2.0.0",
        "Y.Forced.DataType": "boolean",
        "Y.Forced.Value": "False",
        "contextAgent": "VAL OPCUA Pipeline",
        "contextAlgorithm": "VFN direct match",
        "contextConfidenceScore": "1.0",
        "dataSetExternalId": "dataset:opcua_data_val",
        "externalId": "opcua_val_eo:s=b1f559e1-c1b8-40e4-93a1-f750c17a9b72",
        "labels": "label:val-opc-ua",
        "location": "VFN",
        "name": "47-PS-12001-M01-01",
        "parentExternalId": "VFN_47-PS-12001-M01-01",
        "source": "OPC_UA"
    },
    "source": "OPC_UA",
    "labels": [
        {
            "externalId": "label:val-opc-ua"
        }
    ],
    "id": 812863580834318,
    "created_time": "2021-10-27 00:31:34",
    "last_updated_time": "2021-10-27 00:31:34",
    "root_id": 1202379980374596
}
{
    "external_id": "VFN_47-PS-12001-M01-01",
```

```
"name": "47-PS-12001-M01-01",
"parent_id": 6877709216836998,
"parent_external_id": "VFN_47-PS-12001-M01",
"description": "SEA WATER LIFT PUMP, START/STOP",
"data_set_id": 6810015115223364,
"metadata": {
    "AVEVA TAG STATUS": "Active",
    "CALIBRATED RANGE MAX": "",
    "CALIBRATED RANGE MIN": "",
    "CATEGORY CODE": "SG",
    "CPSR CODE 01 (CAUSE)": "",
    "CPSR CODE 02 (EFFECT)": "",
    "CPSR CODE 03 (PURPOSE)": "",
    "DETECTOR RANGE": "",
    "EQUIPMENT IDENTIFIER": "",
    "EQUIPMENT/LINE NUMBER": "",
    "EX RATED EQUIPMENT REGISTER (HAZ AREA) FLAG": "",
    "FACILITY": "VFN",
    "FIRE AREA": "",
    "FUNCTION CODE": "QE",
    "FUNCTION CODE DESCRIPTION": "ELECTRICAL INTERFACE SIGNALS",
    "HAZARDOUS AREA CERT NUMBER": "",
    "HAZARDOUS AREA CERT STANDARD": "",
    "HAZARDOUS AREA GAS GROUP (AS REQUIRED)": "",
    "HAZARDOUS AREA PROTECTION (AS REQUIRED)": "",
    "HAZARDOUS AREA RATING (AS REQUIRED)": "",
    "HAZARDOUS AREA TEMP RATING (AS REQUIRED)": "",
    "HAZARDOUS ATEX MARKING": "",
    "I/O TYPE": "",
    "INTRINSICALLY SAFE FLAG": "",
    "IP RATING": "",
    "LOCATION (FACILITY AREA CODE)": "W40",
    "LOOP ID": "",
    "ORIGINATING CONTRACTOR": "",
    "PARENT TAG": "47-PS-12001-M01",
    "PLATFORM CODE": "VFN",
    "PURCHASE ORDER NO": "",
    "REMARKS": "",
    "SAP CATALOG PROFILE": "CL0000001",
    "SAP EXPORT FLAG": "False",
    "SAP OBJECT TYPE": "CL-SI",
    "SERVICE DESCRIPTION": "SEA WATER LIFT PUMP, START/STOP",
```

```
        "SET POINT": "",
        "SET POINT COMMENT": "",
        "SET POINT CRIT FLAG": "",
        "SIGNAL TYPE": "",
        "SUB SYSTEM NO": "",
        "SYSTEM": "47",
        "TAG DISCIPLINE": "I",
        "TAG NUMBER": "47-PS-12001-M01-01",
        "WEIGHT (DRY)": "",
        "WEIGHT (OPERATING)": "",
        "WORKFLOW STATUS": "In operation",
        "dataSetExternalId": "dataset:aveva-net-assets",
        "labels": "label:aveva-net",
        "objectId": "47-PS-12001-M01-01",
        "source": "aveva",
        "state": "0"
    },
    "source": "aveva",
    "labels": [
        {
            "externalId": "label:aveva-net"
        }
    ],
    "id": 482220262989013,
    "created_time": "2020-04-06 16:33:20",
    "last_updated_time": "2022-12-08 08:49:25",
    "root_id": 1202379980374596
}
{
    "external_id": "VFN_VFN",
    "name": "VFN",
    "description": "",
    "data_set_id": 6810015115223364,
    "metadata": {
        "AVEVA TAG STATUS": "",
        "CATEGORY CODE": "ADM",
        "FACILITY": "VFN",
        "FUNCTION CODE": "SYST",
        "FUNCTION CODE DESCRIPTION": "",
        "LOCATION (FACILITY AREA CODE)": "",
        "ORIGINATING CONTRACTOR": "",
        "PARENT TAG": "",
```

```
        "PLATFORM CODE": "VFN",
        "REMARKS": "",
        "SAP EXPORT FLAG": "",
        "SAP OBJECT TYPE": "",
        "SERVICE DESCRIPTION": "",
        "SUB SYSTEM NO": "",
        "SUPERSEDED BY TAG": "",
        "SYSTEM": "",
        "TAG DISCIPLINE": "",
        "TAG NUMBER": "",
        "WORKFLOW STATUS": "",
        "dataSetExternalId": "dataset:aveva-net-assets",
        "labels": "label:aveva-net",
        "objectId": "VFN",
        "source": "aveva",
        "state": "0"
    },
    "source": "aveva",
    "labels": [
        {
            "externalId": "label:aveva-net"
        }
    ],
    "id": 1202379980374596,
    "created_time": "2020-04-06 16:32:41",
    "last_updated_time": "2022-12-08 08:48:20",
    "root_id": 1202379980374596
}
```

In [22]:

```
desired_states = ['01', '02', '03', '04', '05', '06', '14', '24', '31', '32', '36', '38', '39', '52']
desired_subtag = '01'
for field, comps in handover.main_components.items():
    print(f"\nCDF assets with state subtag '{desired_subtag}' at {field}:")
    for comp in comps:
        for asset in comp.assets:
            if asset.state_subtag == desired_subtag:
                events = asset.parent.events(data_set_ids=[handover.fields[field].data_set_id], sort=['createdTime:asc'], limit=-1
)
                print(f'\n  {asset.name} {comp.description}: {len(events)} events')
                if len(events) < 100:
                    for event in events:
                        severity = int(event.metadata['severity'])
                        if severity > 400 and severity < 450:
                            #print(f'    {event.metadata["time"]} | {event.metadata["source"]} | {severity} | {event.description}
 ({event.id})')
                            print_event(event)
                            #print(event)
```

CDF assets with state subtag '01' at ULA:

  PM-0122A-01 Seal Oil Pump - HP/MP Compr.: 55 events

  PM-0122B-01 Seal Oil Pump - HP/MP Compr.: 55 events

  PM-0121A-01 Lube Oil Pump - HP/MP Compr.: 55 events

  PM-0121B-01 Lube Oil Pump - HP/MP Compr.: 55 events

  PM-1101A-01 Cooling Water Resirc. Pump: 16 events

  PM-1101B-01 Cooling Water Resirc. Pump: 16 events

  CM-1101A-01 Air Compressor: 16 events

  CM-1101B-01 Air Compressor: 16 events

  CM-1101C-01 Air Compressor: 16 events

  Z-0501-01 Fire Pump (P): 48 events
    10/30/2021 1:42:41 PM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 749101471457656
0)
    10/30/2021 1:42:51 PM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 4101346624522472)
    12/4/2021 1:51:13 PM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 401253143255296)
    12/4/2021 1:51:02 PM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 2543581893353432)
    3/28/2022 2:49:16 PM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 6341054805114605)
    3/28/2022 2:49:26 PM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 2278841923690908)
    4/1/2022 7:00:20 AM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 7524964992757129)
    4/1/2022 7:00:30 AM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 3865374785134380)
    4/3/2022 7:04:25 AM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 229926686002907)
    4/3/2022 7:04:15 AM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 3648435215831069)
    4/4/2022 12:42:24 PM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 3137362613409224)
    4/4/2022 12:42:34 PM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 5447329490699558)
    4/11/2022 9:29:06 AM | ULP-Z-0501 | **no object description** | Abnormal condition, | 402 (id: 1616559390252543)
    4/11/2022 9:29:16 AM | ULP-Z-0501 | **no object description** | Normal state, | 402 (id: 3161366329677010)
    5/27/2022 7:54:09 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition, | 402 (id: 4015682026184917)
    5/27/2022 7:54:19 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state, | 402 (id: 7011459649688941)
    6/26/2022 1:58:07 PM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition, | 402 (id: 2036856565308853)
    6/26/2022 1:58:17 PM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state, | 402 (id: 3726013229128949)
    6/26/2022 3:40:11 PM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition, | 402 (id: 500375860795344)

```
    6/26/2022 3:40:21 PM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 3695207982388418)
    8/17/2022 8:56:41 PM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 1390247134031085)
    8/17/2022 8:56:51 PM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 2458936580365501)
    8/22/2022 6:59:20 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 2969474078739957)
    8/22/2022 6:59:30 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 3678516510588174)
    8/22/2022 8:58:51 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 6737071791906620)
    8/22/2022 8:59:01 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 5520227485473630)
    9/20/2022 7:18:26 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 1791293262342792)
    9/20/2022 7:18:36 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 2710671804216190)
    10/4/2022 6:11:29 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 4905422204598393)
    10/4/2022 6:11:39 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 1299728177632809)
    10/29/2022 7:56:57 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 6228412139374482)
    10/29/2022 7:57:07 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 369242586665156)
    11/24/2022 8:17:26 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 2014904027548550)
    11/24/2022 8:17:16 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 4525811717436702)
    12/7/2022 10:45:43 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 7203186392628303)
    12/7/2022 10:45:53 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 6947850395910664)
    1/1/2023 7:30:12 AM | ULP-Z-0501 | P03 FIRE PUMP START | Abnormal condition,  | 402 (id: 3090906291190458)
    1/1/2023 7:30:22 AM | ULP-Z-0501 | P03 FIRE PUMP START | Normal state,  | 402 (id: 5314147534700903)

  Z-0502-01 Fire Pump (D): 66 events
    10/30/2021 1:33:21 PM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 1982714283899275)
    10/30/2021 1:33:11 PM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 277987348730489
0)
    10/30/2021 4:04:02 PM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 5462519402500916)
    10/30/2021 4:03:52 PM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 605750563039861
0)
    10/30/2021 4:31:08 PM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 5027354714498722)
    10/30/2021 4:30:58 PM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 739932897160957
8)
    10/31/2021 12:30:20 AM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 427009416030066
6)
    10/31/2021 12:30:30 AM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 7277334652435901)
    10/31/2021 10:11:46 AM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 889108060420682
7)
    10/31/2021 10:11:56 AM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 2654866240986267)
    10/31/2021 1:15:01 PM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 359613906125981)
    10/31/2021 1:15:12 PM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 594531979393946)
    12/4/2021 1:50:58 PM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 6599626851827896)
    12/4/2021 1:51:08 PM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 979846567857581)
    3/15/2022 4:16:50 AM | ULD-Z-0502 | **no object description** | Abnormal condition,  | 402 (id: 2744685135942602)
    3/15/2022 4:17:00 AM | ULD-Z-0502 | **no object description** | Normal state,  | 402 (id: 1347562667891048)
```

4/11/2022 9:29:01 AM | ULD-Z-0502 | **no object description** | Abnormal condition, | 402 (id: 6142749394273742)
4/11/2022 9:29:11 AM | ULD-Z-0502 | **no object description** | Normal state, | 402 (id: 8280401204424208)
6/18/2022 10:27:30 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 7820765317846358)
6/18/2022 10:27:40 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 1779977794511695)
6/18/2022 4:06:35 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 7213848773081269)
6/18/2022 4:06:45 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 7917871415325985)
6/19/2022 5:28:03 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 49155596964129)
6/19/2022 5:27:53 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 8358307197371642)
6/19/2022 6:24:54 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 3052689287781797)
6/19/2022 6:25:04 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 4337522528907327)
6/19/2022 7:25:46 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 2978567523137997)
6/19/2022 7:25:36 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 5210100767158359)
6/19/2022 10:57:19 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 291119728043779)
6/19/2022 10:57:09 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 643993988255980)
6/19/2022 12:00:32 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 5365886987077956)
6/19/2022 12:00:42 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 6717359195423676)
6/19/2022 1:49:38 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 5099607402413380)
6/19/2022 1:49:48 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 7926311976815857)
6/24/2022 12:15:00 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 5930992994618015)
6/24/2022 12:15:10 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 7399871359872713)
6/26/2022 1:57:57 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 2976548792680046)
6/26/2022 1:58:07 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 6456985916472197)
6/26/2022 3:40:01 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 6407592264590835)
6/26/2022 3:40:11 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 7307821722293566)
7/19/2022 4:24:11 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 7202416843863634)
7/19/2022 4:24:21 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 7453663852063464)
8/8/2022 7:22:51 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 50444521874723)
8/8/2022 7:22:40 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 1249751321199561)
8/22/2022 7:13:14 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 451513212598158)
8/22/2022 7:13:24 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 1861704891989765)
8/23/2022 1:09:49 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 2466469722238316)
8/23/2022 1:09:39 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 3185965102791027)
8/26/2022 9:00:21 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 2859449167975175)
8/26/2022 9:00:31 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 4682592284755417)
8/28/2022 1:13:15 PM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 1166863801664749)
8/28/2022 1:13:25 PM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 3884630383528738)
9/17/2022 6:41:36 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 1492375387548487)
9/17/2022 6:41:46 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 5827376308347159)
9/23/2022 10:02:57 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 4226483857459791)
9/23/2022 10:03:07 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 3723784203018993)
9/23/2022 10:12:02 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state, | 402 (id: 5293872641325346)
9/23/2022 10:11:52 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition, | 402 (id: 5402885980178522)

```
    10/29/2022 7:56:51 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition,  | 402 (id: 810825727321880)
    10/29/2022 7:57:01 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state,  | 402 (id: 6870507409856296)
    11/12/2022 9:38:48 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state,  | 402 (id: 7827462560236904)
    11/12/2022 9:38:38 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition,  | 402 (id: 8243950479318741)
    11/22/2022 4:13:19 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition,  | 402 (id: 2887959180972269)
    11/22/2022 4:13:29 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state,  | 402 (id: 6833688137922189)
    12/7/2022 10:45:23 AM | ULD-Z-0502 | D14 FIRE PUMP START | Abnormal condition,  | 402 (id: 7935581878694015)
    12/7/2022 10:45:33 AM | ULD-Z-0502 | D14 FIRE PUMP START | Normal state,  | 402 (id: 8973496850420441)

  Z-0503-01 Fire Pump (Q): 46 events
    10/30/2021 1:33:01 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 399102391474934
7)
    10/30/2021 1:33:11 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 6339656478245538)
    10/30/2021 4:30:58 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 4846407233194008)
    10/30/2021 4:30:48 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 564210234279159
4)
    10/31/2021 12:30:10 AM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 549148924680989
5)
    10/31/2021 12:30:20 AM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 4022678758656581)
    10/31/2021 10:11:46 AM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 3225377771128451)
    10/31/2021 10:11:36 AM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 370254970328739
2)
    10/31/2021 1:14:51 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 729809303696919
0)
    10/31/2021 1:15:01 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 8112753408515499)
    12/4/2021 1:51:13 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 649392814641060)
    12/4/2021 1:51:23 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 710626676971479)
    2/26/2022 5:15:24 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 2061580763366426)
    2/26/2022 5:15:34 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 4492404871508281)
    2/27/2022 1:14:41 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 7042624050204595)
    2/27/2022 1:14:51 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 7208174348450381)
    3/15/2022 4:17:00 AM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 8694600491299514)
    3/15/2022 4:17:10 AM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 358891261630597)
    3/15/2022 4:49:58 AM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 6088664653937912)
    3/15/2022 4:50:08 AM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 6420972509609640)
    3/15/2022 5:04:02 AM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 1812542691148025)
    3/15/2022 5:03:52 AM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 5613005525403526)
    3/28/2022 2:49:26 PM | ULQ-Z-0503 | **no object description** | Abnormal condition,  | 402 (id: 811463024827335)
    3/28/2022 2:49:37 PM | ULQ-Z-0503 | **no object description** | Normal state,  | 402 (id: 8765199515238515)
    5/12/2022 3:00:25 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 2930784974908004)
    5/12/2022 3:00:35 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 515346018540102)
    5/27/2022 7:53:59 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 2949925932061948)
```

```
    5/27/2022 7:54:09 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 6811589863680946)
    7/24/2022 6:41:54 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 7824670960346740)
    7/24/2022 6:42:05 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 8390265023945665)
    8/22/2022 6:59:30 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 2193838411364113)
    8/22/2022 6:59:40 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 4139482645508128)
    8/23/2022 1:09:49 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 8387045572720919)
    8/23/2022 1:09:59 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 2702674262895116)
    8/26/2022 9:00:31 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 2122102038615611)
    8/26/2022 9:00:41 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 8416804166482087)
    8/28/2022 1:13:25 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 5256808648158670)
    8/28/2022 1:13:36 PM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 7816910659693737)
    11/12/2022 9:38:54 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 2643349910267468)
    11/12/2022 9:38:43 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 3015767689990037)
    11/22/2022 4:13:29 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 879887219426413)
    11/22/2022 4:13:39 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 4899140535289764)
    11/24/2022 8:17:16 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 6415247786260564)
    11/24/2022 8:17:06 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 7365324268371683)
    12/7/2022 10:45:33 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Abnormal condition,  | 402 (id: 1805000659270025)
    12/7/2022 10:45:43 AM | ULQ-Z-0503 | Q15 FIRE PUMP START | Normal state,  | 402 (id: 6609958282609840)

  PM-0201A-01 Seawater lift Pump: 16 events

  PM-0201B-01 Seawater lift Pump: 16 events

  PM-0202-01 Seawater lift Pump Q: 36 events

  PM-1202A-01 Diesel Forw. Pump: 20 events

  PM-1202B-01 Diesel Forw. Pump: 20 events

CDF assets with state subtag '01' at VAL:

  80-PA-8015A-M01_01 Prod.water booster pump (IP): 0 events

  80-PA-8015B-M01_01 Prod.water booster pump (IP): 0 events

  47-PS-12001-M01-01 Seawaterlift pump: 86 events
    1/27/2022 10:08:04 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal,  | 409 (id: 2592908100484675)
    1/27/2022 10:08:09 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm,  | 409 (id: 5061203643003109)
    1/27/2022 10:08:14 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal,  | 409 (id: 5651269445276220)
    1/27/2022 5:25:58 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm,  | 409 (id: 5341529723675945)
    2/10/2022 10:55:25 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal,  | 409 (id: 1819065654825073)
```

2/10/2022 10:58:25 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 6204901128190573)
4/14/2022 3:39:05 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 6465401880316161)
4/14/2022 3:39:10 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 2833134506769352)
4/14/2022 3:39:15 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 8618014275890515)
4/14/2022 3:44:45 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 5969822959479217)
4/22/2022 8:30:42 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 536408746374050)
4/22/2022 8:30:52 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 6643628593728754)
4/22/2022 8:30:47 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 7546134628299187)
4/22/2022 8:34:42 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 5151416694194626)
5/24/2022 2:14:17 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 3260781002199799)
5/24/2022 2:42:16 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 6766435362519797)
6/6/2022 7:06:43 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 1158588586500001)
6/8/2022 8:35:11 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 632196374982930)
6/8/2022 11:58:21 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 7382906523898058)
6/17/2022 5:57:12 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 7897918140996603)
6/17/2022 6:11:17 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 7182147181807511)
6/17/2022 6:39:22 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 1485544227160558)
6/17/2022 6:46:17 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 440274049972281)
6/17/2022 7:14:32 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 4415545061368277)
6/18/2022 1:51:46 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 2785965218788990)
6/21/2022 4:55:59 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 7653718327696344)
6/25/2022 4:19:26 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 8021780492594301)
6/25/2022 4:21:46 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 1881999431370219)
7/5/2022 3:33:00 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 6199332581111601)
7/5/2022 4:03:45 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 8875188514814331)
2022-10-03T15:21:17.3678Z | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 1802689399398129)
10/11/2022 11:52:06 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 7050245086079525)
10/13/2022 7:06:50 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 7291481119526142)
10/13/2022 11:08:20 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 6736065563728904)
10/25/2022 6:41:32 AM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 2424420651778578)
12/18/2022 3:40:40 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Normal, | 409 (id: 1051701327546854)
12/18/2022 3:46:30 PM | VFN-47-PS-12001 | FN SeawaterLiftPump | Alarm, | 409 (id: 2668564539905920)

47-PS-12001-M01_01 Seawaterlift pump: 42 events
1/27/2022 10:08:02 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 2170320108350359)
1/27/2022 5:25:45 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 1857648793667427)
2/10/2022 10:55:18 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 7801909803982919)
2/10/2022 10:58:16 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 876861988400262)
4/14/2022 3:39:05 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 87405260052825)
4/14/2022 3:44:35 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 3806159206259658)
4/22/2022 8:30:41 PM | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 1036636015596108)
4/22/2022 8:34:30 PM | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 8076814299171252)

5/24/2022 2:14:15 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 3560177418238792)
5/24/2022 2:42:02 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 1498433474449412)
6/6/2022 7:06:28 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 333667839873956)
6/8/2022 8:35:09 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 4309752407473199)
6/8/2022 10:43:54 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 8643209655342483)
6/8/2022 10:59:53 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 3812365427391662)
6/8/2022 10:59:44 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 4550890096306879)
6/8/2022 2:02:46 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 191985517509905)
6/17/2022 5:56:59 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 206123216039280)
6/17/2022 6:11:06 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 2791838086699190)
6/17/2022 6:39:13 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 179730275020369)
6/17/2022 6:46:06 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 3635014732333060)
6/17/2022 7:14:23 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 3071293271408193)
6/18/2022 1:51:37 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 820517571124674)
6/21/2022 4:55:46 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 8988978431828434)
6/25/2022 7:47:13 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 6410173482032673)
6/25/2022 4:21:33 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 8147940094045364)
7/5/2022 3:32:50 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 2678574681272561)
7/5/2022 4:03:33 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 3587348790410112)
2022-10-03T15:04:46.5528Z | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 583310642859691)
2022-10-03T15:05:07.0528Z | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 1997007344387849)
2022-10-03T15:21:08.0528Z | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 180940314418897)
2022-10-03T15:19:24.5528Z | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 475140383518600)
2022-10-03T15:18:05.5528Z | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 3271147193281032)
2022-10-03T15:19:04.0528Z | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 3386488213329783)
2022-10-03T15:17:45.0528Z | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 7346701373513210)
2022-10-03T14:58:35.0528Z | VFN-47-PS-12001 | **no object description** | Alarm, | 409 (id: 2182628398026982)
2022-10-03T14:58:14.5528Z | VFN-47-PS-12001 | **no object description** | Normal, | 409 (id: 5115669233418018)
10/11/2022 11:51:54 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 7456780705443029)
10/13/2022 7:06:51 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 1482256141258748)
10/13/2022 7:07:12 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 7681384081040665)
10/25/2022 6:41:21 AM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 7429615303601381)
12/18/2022 3:39:17 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Alarm, | 409 (id: 2439567122404035)
12/18/2022 3:46:21 PM | VFN-47-PS-12001 | FN Seaw.P.Start/Stop | Normal, | 409 (id: 4385391287491855)

47-PS-11001-M01_01 Seawaterlift pump: 58 events
5/28/2022 9:31:38 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 1530258452846246)
5/28/2022 9:31:43 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 7921936603595858)
5/28/2022 10:03:35 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 8974500542703015)
5/28/2022 10:03:38 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 3767365978517893)
5/28/2022 10:03:55 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 5265811023192755)
5/28/2022 10:03:58 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 6336435807846601)

5/29/2022 5:36:14 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 1182881894447231)
5/29/2022 5:36:18 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 907700922058946)
5/29/2022 5:36:37 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 1113131637761335)
5/29/2022 5:36:34 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 2244333788953995)
5/30/2022 5:37:56 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 2087720478657923)
5/30/2022 5:38:14 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 2648986654531983)
5/30/2022 5:38:16 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 7068869748018743)
5/30/2022 5:37:54 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 7581045729768805)
5/31/2022 5:39:23 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 7695988742427235)
5/31/2022 5:39:26 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 8496878557323042)
5/31/2022 5:39:46 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 1052213990418152)
5/31/2022 5:39:43 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 3557048607106753)
5/31/2022 12:43:25 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 3810085311099335)
5/31/2022 12:43:21 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 4812419290407376)
5/31/2022 12:43:19 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 6256510130814025)
5/31/2022 12:43:31 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 4575333393304352)
6/1/2022 5:36:30 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 5337239126851783)
6/1/2022 5:36:27 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 8790979946615093)
6/1/2022 5:36:47 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 885901609513632)
6/1/2022 5:36:50 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 3191482705367877)
6/2/2022 5:40:42 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 4828121947026946)
6/2/2022 5:40:44 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 159204452756260)
6/2/2022 5:41:02 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 1883856426236850)
6/2/2022 5:41:04 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 5207466875697522)
6/3/2022 5:39:26 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 4422960486610548)
6/3/2022 5:39:29 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 2069349905592225)
6/3/2022 5:39:46 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 2370583274238058)
6/3/2022 5:39:49 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 7505421001111071)
6/4/2022 5:41:05 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 3107004291953154)
6/4/2022 5:41:08 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 3278320034972841)
6/4/2022 5:41:25 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 6406209566514150)
6/4/2022 5:41:28 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 3231353036295970)
6/4/2022 4:13:58 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 1406798475761791)
6/4/2022 4:13:55 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 6251132864069939)
6/10/2022 6:21:04 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 593252636577309)
6/10/2022 6:21:09 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 3403045454230413)
6/13/2022 5:56:52 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 4765900997050670)
6/13/2022 5:56:57 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 7482815598077465)
6/25/2022 7:47:06 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 2234233888732857)
6/25/2022 7:47:11 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm, | 409 (id: 8175508716545156)
6/25/2022 3:12:18 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 5810981656668012)
6/25/2022 3:12:21 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal, | 409 (id: 8698023624395267)

```
6/28/2022 7:12:50 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm,  | 409 (id: 8725845405774338)
6/28/2022 7:12:54 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm,  | 409 (id: 4698050576471597)
6/28/2022 8:50:35 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal,  | 409 (id: 6052060068591876)
6/28/2022 8:50:39 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal,  | 409 (id: 8225144357525869)
7/5/2022 3:31:59 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm,  | 409 (id: 491384756583602)
7/5/2022 3:31:56 AM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Alarm,  | 409 (id: 8996359027053760)
7/5/2022 4:02:48 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal,  | 409 (id: 2605179207072769)
7/5/2022 4:02:49 PM | VFS-47-PS-11001 | FS Seaw.P.Start/Stop | Normal,  | 409 (id: 6184515918168092)
2022-10-03T12:49:34.6447Z | VFS-47-PS-11001 | **no object description** | Normal,  | 409 (id: 1511968324727463)
2022-10-03T12:49:32.6832Z | VFS-47-PS-11001 | **no object description** | Normal,  | 409 (id: 2628377366410730)
```

Example component P-04103 at Ula and its found assets in the OPC UA dataset:

Assets	Exact Match
P-04103_27	No
P-04103_08	No
P-04103-08	No
P-04103	**Yes**
P-04103-09	No

Write overviews of found components and assets

Overview of all OPC UA assets:

In [23]:

```
with open('output/all_assets_ula.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(['ASSET'])
    for asset in handover.fields['ULA'].assets:
        row = [asset.name]
        writer.writerow(row)
```

Overview of all components and which ones that had assets related to them:

In [24]:

```
with open('output/found_components_ula.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(['TAG', 'FOUND', 'EXACT'])
    for comp in handover.fields['ULA'].main_components:
        row = [comp.name, 'No', 'No']
        if comp.assets: row[1] = 'Yes'
        if comp.exact_match: row[2] = 'Yes'
        writer.writerow(row)
        #print(row)
```

Overview of all found assets and their corresponding main component tag:

In [25]:

```
content = [[]]
for comp in handover.main_components['ULA']:
    content[0].append(comp.name)
    content.append([])
    for asset in comp.assets:
        content[-1].append(asset.name)
#print(content)
with open('output/found_assets_ula.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(content[0])
    for i in range(max([len(l) for l in content[1:]])):
        row = []
        for j in range(len(content[0])):
            if len(content[j+1]) > i:
                row.append(content[j+1][i])
            else:
                row.append('')
        writer.writerow(row)
```

Map event severities

Ref. Alarm FDS documents G00-23-IN-5067-00 and VAL-AC-I-0212

In [26]:

```
systems = {
    'ULA': {
        '01': 'ESD',
        '02': 'FnG',
        '03': 'PSD',
        '04': 'PCS',
        '09': 'AC400'
    },
    'VAL': {
        '01': 'ESD',
        '02': 'FnG',
        '03': 'PSD',
        '04': 'PCS',
        '05': 'HVAC',
        '06': 'SCMS',
        '09': 'AC400'
    }
}
priorities = {
    '9': 'Pri.1',
    '8': 'Pri.2',
    '7': 'Pri.3',
    '6': 'Pri.4',
    '4': 'EVENT'
}
```

The following cell goes through all events from the found assets and maps their severities:

In [27]:

```
today = datetime.datetime.today()
shift_start = datetime.datetime(today.year, today.month, today.day, 7)
shift_end = shift_start + datetime.timedelta(hours=12)
divisor = timestamp_to_ms(shift_end) - timestamp_to_ms(shift_start)
epoch_correction = timestamp_to_ms(shift_start) - timestamp_to_ms(datetime.datetime(today.year, today.month, today.day))
max_bin = math.floor((timestamp_to_ms(shift_start) - epoch_correction) / divisor)

field_metadata = {}
metadata_mask = ['Emitter', 'source', 'SourceNode', 'id', 'activeTime', 'SourceName', 'eventCounter', 'time', 'objectDescription',
'ObjectDescription', 'EventData', 'message']

severity_mapping = {}

alarm_states = {
    'ULA': {
        'alarmStates': [None],
        'newStates': [None],
        'stateMap': [[0]]
    },
    'VAL': {
        'alarmStates': [None],
        'newStates': [None],
        'stateMap': [[0]]
    }
}

# Loop through all fields (Aker BP 'assets'/installations, e.g. Ula, Valhall, etc.)
for field, comps in handover.main_components.items():
    print(f'Mapping alarm & event severities on {handover.fields[field].name}...')
    time_ref = datetime.datetime.now()

    metadata = {'all': {}, 'common': {'alarms': None, 'events': None, 'other': None}, 'irregular': {'alarms': [], 'events': [], 'o
ther': []}}
    event_count = {'alarms': 0, 'events': 0, 'other': 0}

    severities = {}
    n_events = 0
    n_shifts = 0
```

```
    new_state_index = {}
    alarm_state_index = {}
    alarm_states[field]['severities'] = {}

    # Loop through all main components for the respective field
    for comp_index, comp in enumerate(comps):

        # Loop through all CDF assets found from the respective main component tag
        for asset in comp.assets:
            asset_changed = True

            # Download all (limit=-1) events from the given CDF asset that is part of the given fields dataset, sorted by 'created
Time' in ascending order
            events = asset.parent.events(data_set_ids=[handover.fields[field].data_set_id], sort=['createdTime:asc'], limit=-1)

            # Loop through all events found from the respective CDF asset
            for event in events:
                n_events += 1

                # Prints progress percentages every 5 seconds while cell is running
                if datetime.datetime.now() - time_ref > datetime.timedelta(seconds=5):
                    if asset_changed:
                        print(f'{comp_index/len(comps)*100:.1f}%')
                        asset_changed = False
                    else:
                        print(f'{comp_index/len(comps)*100:.1f}% - {asset.name}')
                    time_ref = datetime.datetime.now()

                # Determine SAS event type, i.e. alarm or event
                event_type = None
                if int(event.metadata['severity']) > 600 and int(event.metadata['severity'])%50 != 0:
                    event_type = 'alarms'
                elif int(event.metadata['severity']) > 400 and int(event.metadata['severity']) < 450:
                    event_type = 'events'
                else:
                    event_type = 'other'
                event_count[event_type] += 1

                # Metadata mapping
                if metadata['common'][event_type] == None: metadata['common'][event_type] = list(event.metadata.keys())
                for data, value in event.metadata.items():
                    if data not in metadata['all']: metadata['all'][data] = {'values': [], 'alarms': 0, 'events': 0, 'other': 0}
```

```
        metadata['all'][data][event_type] += 1
        if data not in metadata_mask:
            if value not in metadata['all'][data]['values']: metadata['all'][data]['values'].append(value)
        if data not in metadata['common'][event_type]:
            if data not in metadata['irregular'][event_type]: metadata['irregular'][event_type].append(data)
    for data in metadata['common'][event_type]:
        if data not in event.metadata:
            if data not in metadata['irregular'][event_type]: metadata['irregular'][event_type].append(data)
            metadata['common'][event_type].remove(data)

    # Severity mapping
    severity = event.metadata['severity']
    if severity not in severities: severities[severity] = []
    bin_index = max_bin - math.floor((event.last_updated_time - epoch_correction) / divisor)
    while len(severities[severity]) <= bin_index: severities[severity].append(0)
    severities[severity][bin_index] += 1
    if len(severities[severity]) > n_shifts: n_shifts = len(severities[severity])

    # Alarm mapping
    new_state_found = None
    alarm_state_found = None
    if 'newState' in event.metadata:
        new_state_found = event.metadata['newState']
        if new_state_found not in alarm_states[field]['newStates']:
            new_state_index[new_state_found] = len(alarm_states[field]['newStates'])
            alarm_states[field]['newStates'].append(new_state_found)
            alarm_states[field]['stateMap'].append([0 for i in range(len(alarm_states[field]['stateMap'][-1]))])
    if 'AlarmState' in event.metadata:
        alarm_state_found = event.metadata['AlarmState']
        if event.metadata['AlarmState'] not in alarm_states[field]['alarmStates']:
            alarm_state_index[alarm_state_found] = len(alarm_states[field]['alarmStates'])
            alarm_states[field]['alarmStates'].append(alarm_state_found)
            for row in alarm_states[field]['stateMap']:
                while len(row) < len(alarm_states[field]['alarmStates']): row.append(0)
    elif 'alarmState' in event.metadata:
        alarm_state_found = event.metadata['alarmState']
        if alarm_state_found not in alarm_states[field]['alarmStates']:
            alarm_state_index[alarm_state_found] = len(alarm_states[field]['alarmStates'])
            alarm_states[field]['alarmStates'].append(alarm_state_found)
            for row in alarm_states[field]['stateMap']:
                while len(row) < len(alarm_states[field]['alarmStates']): row.append(0)
    if new_state_found or alarm_state_found:
```

```
                    alarm_states[field]['stateMap'][new_state_index[new_state_found] if new_state_found else 0][alarm_state_index[
alarm_state_found] if alarm_state_found else 0] += 1
                    if severity not in alarm_states[field]['severities']: alarm_states[field]['severities'][severity] = []
                    while len(alarm_states[field]['severities'][severity]) <= bin_index: alarm_states[field]['severities'][severit
y].append([])
                    alarm_states[field]['severities'][severity][bin_index].append(Alarm(event.id, new_state_found, alarm_state_fou
nd))

                # Event mapping
                if event_type == 'events':
                    pass

    field_metadata[field] = metadata
    field_metadata[field]['event_count'] = event_count

    severity_mapping[field] = severities
    print(f'Done - Mapped {n_events} alarms & events over {n_shifts} shifts:\n')

    severity_dict = {sys: {pri: 0 for pri in priorities} for sys in systems[field]}
    for severity, bins in severities.items():
        total = 0
        for b in bins:
            total += b
        sys = severity[1:]
        pri = severity[0]
        if sys not in severity_dict: severity_dict[sys] = {priority: 0 for priority in severity_dict['01']}
        if pri in severity_dict[sys]:
            severity_dict[sys][pri] = total
        else:
            for system in severity_dict:
                severity_dict[system][pri] = total if system == sys else 0
    mat = []
    for entry in severity_dict.values():
        mat.append([])
        for count in entry.values():
            mat[-1].append(count)
    col_headers = [priorities[key] if key in priorities else f"'{key}'" for key in severity_dict['01']]
    row_headers = [systems[field][key] if key in systems[field] else f"'{key}'" for key in severity_dict]
    print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n')
```

```
Mapping alarm & event severities on Ula...
7.3%
22.0%
36.6%
43.9%
56.1%
87.8%
Done - Mapped 28646 alarms & events over 879 shifts:

       Pri.1  Pri.2  Pri.3  Pri.4  EVENT   '5'   '2'
ESD        0      0      0      0      0     0     0
FnG      758      0   1751      0   2893     0     0
PSD        0      0      0      0      0     0     0
PCS        0    537   1281   1169   4767     0     0
AC400      0     93   1805     31   4090     0     0
'00'       0      0      0     39      3  1531  7898

Mapping alarm & event severities on Valhall...
0.0%
2.9%
2.9%
8.6%
14.3%
17.1%
22.9%
25.7%
28.6%
28.6%
31.4%
31.4%
34.3%
34.3%
42.9%
48.6%
51.4%
60.0%
62.9%
62.9%
68.6%
74.3%
77.1%
```

```
77.1%
88.6%
Done - Mapped 18798 alarms & events over 879 shifts:
```

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'2'	'5'	'1'
ESD	19	0	0	0	189	0	0	0
FnG	10	63	0	0	186	0	0	0
PSD	0	21	285	98	1683	0	0	0
PCS	149	0	2662	99	3810	0	0	0
HVAC	0	0	0	0	0	0	0	0
SCMS	0	56	106	491	659	0	0	0
AC400	6	32	410	0	4144	0	0	0
'00'	31	0	39	48	0	2156	514	43
'50'	0	0	0	0	0	0	789	0

The results from the cell above shows a huge amount of alarms & events related to the assets that were found. This shows that reducing the focus to alarms & events only from the main components is not enough, and that further reduction(/filtration) is neccessary in hope of an acceptable amount of alarms & events.

In [28]:

```
metadata_irregularities = {}
for field, metadata in field_metadata.items():
    #print(f'\nMetadata found at {field}:')
    #print(f'  Common attributes for alarms at {field}:\n  {metadata["common"]["alarms"]}\n')
    #print(f'  Irregular attributes for alarms at {field}:\n  {metadata["irregular"]["alarms"]}\n')
    #print(f'  Common attributes for events at {field}:\n  {metadata["common"]["events"]}\n')
    #print(f'  Irregular attributes for events at {field}:\n  {metadata["irregular"]["events"]}\n')

    for event_type, data in metadata['irregular'].items():
        if event_type != 'other':
            for attr in data:
                if attr not in metadata_irregularities: metadata_irregularities[attr] = {}
                if event_type not in metadata_irregularities[attr]: metadata_irregularities[attr][event_type] = {'TOT': 0}
                metadata_irregularities[attr][event_type][field] = metadata['all'][attr][event_type]/metadata['event_count'][event
_type]*100

    #for attr, data in metadata['all'].items():
    #    percentages = [f'{data[key]/metadata["event_count"][key]*100:.1f}' for key in data if key != 'values' and key != 'other']
    #    print(f'\n"{attr}": {percentages}')
        #if data['values']: print(data['values'])
    print('\n')

#for attr, event_types in metadata_irregularities.items():
#    print(f'{attr}:')
#    for event_type, ps in event_types.items():
#        print(f'  {event_type}: {ps}')
```

Below is a table showing all metadata attributes found on alarms and events in CDF with example values, that we can use to filter the data to further reduce the amount. The 'alarm' and 'event' columns shows the percentage of alarms or events that has that has that specific metadata attribute. E.g. the ackRequired attribute is found on 61,5% of the alarms from Ula, but only on 7,4% of the alarms from Valhall. No events (from neither Ula nor Valhall) was found with this attribute. 100% means that all alarms/events were found with that specific attribute, on both Ula and Valhall.

metadata	alarm (ULA/VAL)	event (ULA/VAL)	value (e.g.)	description	of interest
ackRequired	61,5%/7,4%		'True', 'False'		Maybe
activeTime	61,5%/7,4%		'5/24/2022 5:22:01 PM'		
actorId	61,8%/7,4%	~0,1%/--~	'ULA\eivhog', 'ULA\operator', ...	Probably shows who did some action related to the event	Maybe
alarmState	**~0,5%/0,1%~**		**'RTN', 'ACT', 'ABL'**	**Return-to-normal, Active, Automatically Blocked**	Yes
AlarmState	**61,0%/6,6%**		**'ACT', 'RTN', 'ABL', 'SLV'**	**Return-to-normal, Active, Automatically Blocked, Shelved??**	Yes
changeMask	61,5%/7,4%		'243', '2', '193', '1', '195', ...	Assumed correlation with 'changeMaskText'	?
changeMaskText	61,5%/7,4%		'Active, Ack, Severity, ...', ...	Assumed correlation with 'changeMask', ...	
class	~0,5%/0,2%~	~0,1%/0,2%~	'0', '2002', '4006', '8', ...		
Class	76,1%/65,8%	97,2%/56,0%	'0', '79', '2002', '127', ...		?
conditionName	61,5%/7,4%		'Sig.Err', 'Value', 'Low Low', ...		Maybe
subConditionName	61,5%/7,4%		'Sig.Err', 'Value', 'Low Low'		Maybe
contextAgent	100%	100%	'OPC UA contextualization pipeline'	Related to contextualization into CDF	
contextAlgorithm	100%	100%	'Exact match - Primary location', ...	Related to contextualization into CDF	
contextConfidenceScore	100%	100%	'1'	Related to contextualization into CDF	
Emitter	100%	100%	'opcua_val_eo:s=e102c1d1-fcbf...', ...	Related to data transfer	
eventCategory	100%	100%	'1258335080', '617833666', ...	Assumed correlation with 'eventCategoryName'	
eventCategoryName	100%	100%	'SimpleProcess(MB300)', ...	Assumed correlation with 'eventCategory'	
eventCounter	100%	100%	'86749552', ...	Probably an incrementing number as events accumulate	
EventData	100%	100%	string of metadata	A duplicate of the events metadata	
eventType	100%	100%	'Simple', 'Tracking', 'Condition'		?
hidden	~0,5%/0,1%~		'False'		
Hidden	61,0%/6,6%		'False'		
id	100%	100%	'4974afd8-ad9d-4a35-9058...', ...	Probably the alarm or events id in the control system	
message	**100%**	**100%**	**'Fault', ...**	**Event message**	Yes
newState	**61,5%/7,4%**		**'1', '3', '5', '7'**	**Correlating with 'newStateText'**	Yes

metadata	alarm (ULA/VAL)	event (ULA/VAL)	value (e.g.)	description	of interest
newStateText	61,5%/7,4%		'Enabled', 'Enabled, Active', ...	Correlating with 'newState'	
objectDescription	~0,5%/0,2%~	~0,1%/0,2%~	'PH LQ Gas Mech Equip L135B', ...		
ObjectDescription	**76,1%/65,8%**	**97,2%/56,0%**	**'PH LQ Gas Mech Equip L135B', ...**		**Yes**
processSection	~0,1%/0,0%~	~0,1%/0,1%~	'2', '4', '10', '13', '8', ...		
ProcessSection	26,3%/0,4%	94,7%/7,7%	'2', '4', '1', '10', '8', ...		?
quality	61,5%/7,4%		'0', '2147483648'	Assumed correlation with 'qualityText'	?
qualityText	61,5%/7,4%		'Good', 'Bad'	Assumed correlation with 'quality'	
severity	**100%**	**100%**	**'409', '200', '709', '809', ...**	**First digit states the alarm priority, last digits state the system**	Yes
Severity	**100%**	**100%**	**'409', '200', '709', '809', ...**	**First digit states the alarm priority, last digits state the system**	Yes
source	**100%**	**100%**	**'49-FE-G-P111-002', ...**	**System tag number of event source**	Yes
SourceName	100%	100%	'49-FE-G-P111-002', ...	Assumed duplicate of *source*	
SourceNode	100%	100%	'opcua_val_eo:s=e102c1d1-fcbf...', ...	Related to data transfer	
time	**100%**	**100%**	**'5/24/2022 6:25:12 PM', ...**	**Time of event**	Yes
type	100%	100%	'opcae'		
version	100%	100%	'2.0.0'		

By examining the metadata of all alarms & events we find that many alarms have properties that are unique to alarms, e.g. `alarmState` and `newState`, telling us something about the state of the alarm. The state of an alarm will change e.g. when the situation that resulted in an alarm returns to normal and/or the alarm is acknowledged. The table below shows what the states of the alarms is telling us. I assume that an alarm corresponding to a real world event/situation can move between states (e.g. from `3` to `7` when it gets acknowledged), resulting in multiple alarms related to the same real world event/situation, and that new alarms (before they either return-to-normal or gets acknowledged) should have `newState = 3`.

Shelved	Hidden	?	Ackn.	Active	Enable	newState	newStateText
0	0	0	0	0	0	0	
0	0	0	0	0	1	1	Enabled
0	0	0	0	1	0	2	
0	0	0	0	1	1	**3**	**Enabled, Active**
0	0	0	1	0	0	4	
0	0	0	1	0	1	5	Enabled, Acknowledged
0	0	0	1	1	0	6	
0	0	0	1	1	1	7	Enabled, Active, Acknowledged
...	...	...	...	...	...	...	
bit5	bit4	bit3	bit2	bit1	bit0	value	

The following cell maps `newState` against `alarmState`, where `alarmState` also gives an description of what state the alarm has.

alarmState	Description
ACT	Active
RTN	Return-to-normal
ABL	Automatically blocked
SLV	Shelved?

In [29]:

```
for field, data in alarm_states.items():
    ae_map = data['stateMap']
    col_headers = data['alarmStates']
    row_headers = data['newStates']
    print(f'Mapping of alarms that has a newState or alarmState property for {field}:')
    print(f'{pd.DataFrame(ae_map, columns=col_headers, index=row_headers)}\n')
```

```
Mapping of alarms that has a newState or alarmState property for ULA:
     NaN   RTN   ACT  ABL  SLV
NaN    0     0     0    0    0
1      2  1071     0  169    6
3      2     0  1902   26   12
7     11     0  1564   10    3
5      0  1542     0   20    3

Mapping of alarms that has a newState or alarmState property for VAL:
     NaN  RTN  ACT  ABL
NaN    0    0    0    0
1    446  694    0    3
3    143    0  629   16
5     94  545    0   10
7     60    0  521   13
```

The results from the mapping above confirms that active alarms (ACT) can only have a `newState` value of `3` or `7`, and that alarms that has returned to normal (RTN) and are no longer active can only have `1` or `5`, which corresponds with the table of `newState` values above. However, we see that all `newState` values can also have ABL and SLV as their `alarmState`. I would assume that alarms that is automatically blocked (ABL), from the control system I guess, will not be of any interest for the operators and their handover, but if SLV means that the alarm is shelved, those alarms could be of interest, e.g. to remind someone that the alarm is in fact shelved. We can also see that on Valhall there are a lot of alarms with the `newState` property that has no `alarmState` property. All alarms that has an `alarmState` does, however, have a `newState`, on both Ula and Valhall.

With a `newState` value of `3` and either `alarmState ACT` or no `alarmState` (for Valhall) we find the following alarms and events:

In [30]:

```
# Settings
desired_n_shifts = None
desired_new_states = ['3']
desired_alarm_states = ['ACT', None]

# Print severity mapping
for field, data in alarm_states.items():
    n_shifts = 0
    severity_dict = {sys: {pri: 0 for pri in priorities} for sys in systems[field]}
    found_shifts = {'tot': []}
    for severity, bins in data['severities'].items():
        n_shifts = max(n_shifts, len(bins))
        sys = severity[1:]
        pri = severity[0]
        total = 0
        for shift_n, b in enumerate(bins[:desired_n_shifts]):
            for alarm in b:
                if alarm.new_state in desired_new_states and alarm.alarm_state in desired_alarm_states:
                    total += 1
                    if sys in ['01', '02', '03', '04', '05', '06', '09'] and pri in ['9', '8']:
                        if sys not in found_shifts: found_shifts[sys] = []
                        if pri not in found_shifts: found_shifts[pri] = []
                        if shift_n not in found_shifts[sys]: found_shifts[sys].append(shift_n)
                        if shift_n not in found_shifts[pri]: found_shifts[pri].append(shift_n)
                        if shift_n not in found_shifts['tot']: found_shifts['tot'].append(shift_n)
                    #break
        if sys not in severity_dict: severity_dict[sys] = {priority: 0 for priority in severity_dict['01']}
        if pri in severity_dict[sys]:
            severity_dict[sys][pri] = total
        else:
            for system in severity_dict:
                severity_dict[system][pri] = total if system == sys else 0
    print(f'\nMapping of alarms & events at {field} over {desired_n_shifts if desired_n_shifts else "all"} out of {n_shifts} shift
s:\n')
    mat = []
    for entry in severity_dict.values():
        mat.append([])
        for count in entry.values():
            mat[-1].append(count)
```

```
    col_headers = [priorities[key] if key in priorities else f"'{key}'" for key in severity_dict['01']]
    row_headers = [systems[field][key] if key in systems[field] else f"'{key}'" for key in severity_dict]
    print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n')
    for key, values in found_shifts.items():
        print(f'{key}: {len(values)}')
```

Mapping of alarms & events at ULA over all out of 879 shifts:

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'5'
ESD	0	0	0	0	0	0
FnG	152	0	355	0	0	0
PSD	0	0	0	0	0	0
PCS	0	177	133	225	0	0
AC400	0	29	586	0	0	0
'00'	0	0	0	6	0	241

tot: 134
04: 14
8: 32
02: 114
9: 114
09: 18

Mapping of alarms & events at VAL over all out of 879 shifts:

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'5'
ESD	3	0	0	0	0	0
FnG	2	0	0	0	0	0
PSD	0	3	48	12	0	0
PCS	27	0	344	22	0	0
HVAC	0	0	0	0	0	0
SCMS	0	20	12	116	0	0
AC400	2	10	124	0	0	0
'00'	2	0	7	0	0	16
'50'	0	0	0	0	0	2

tot: 36
06: 3
8: 15
09: 12
9: 22
03: 3
01: 1
02: 1
04: 18

We can see that the amount of alarms are reduced considerably if we use a `newState` and `alarmState` filter.

The following cell maps the `newState` and `alarmState` properties with alarm priority and system.

In [31]:

```
for field, data in alarm_states.items():
    state_mapping = {
        'priorities': {f'{i}': {pri: 0 for pri in priorities} for i in range(1,8,2)},
        'systems': {sys: {f'{i}': 0 for i in range(1,8,2)} for sys in systems[field]}
    }
    alarm_mapping = {
        'priorities': {None: {pri: 0 for pri in priorities}},
        'systems': {sys: {None: 0} for sys in systems[field]}
    }
    for severity, bins in data['severities'].items():
        sys = severity[1:]
        pri = severity[0]
        total = 0
        for b in bins:
            for alarm in b:
                if alarm.new_state not in state_mapping['priorities']: state_mapping['priorities'][alarm.new_state] = {priority: 0
for priority in state_mapping['priorities']['1']}
                if pri in state_mapping['priorities'][alarm.new_state]:
                    state_mapping['priorities'][alarm.new_state][pri] += 1
                else:
                    for s in state_mapping['priorities']:
                        state_mapping['priorities'][s][pri] = 1 if s == alarm.new_state else 0
                if sys not in state_mapping['systems']: state_mapping['systems'][sys] = {s: 0 for s in state_mapping['systems']['0
1']}
                if alarm.new_state in state_mapping['systems'][sys]:
                    state_mapping['systems'][sys][alarm.new_state] += 1
                else:
                    for system in state_mapping['systems']:
                        state_mapping['systems'][system][alarm.new_state] = 1 if system == sys else 0

                if alarm.alarm_state not in alarm_mapping['priorities']: alarm_mapping['priorities'][alarm.alarm_state] = {priorit
y: 0 for priority in alarm_mapping['priorities'][None]}
                if pri in alarm_mapping['priorities'][alarm.alarm_state]:
                    alarm_mapping['priorities'][alarm.alarm_state][pri] += 1
                else:
                    for a in alarm_mapping['priorities']:
                        alarm_mapping['priorities'][a][pri] = 1 if a == alarm.alarm_state else 0
                if sys not in alarm_mapping['systems']: alarm_mapping['systems'][sys] = {s: 0 for s in alarm_mapping['systems']['0
1']}
```

```
            if alarm.alarm_state in alarm_mapping['systems'][sys]:
                alarm_mapping['systems'][sys][alarm.alarm_state] += 1
            else:
                for system in alarm_mapping['systems']:
                    alarm_mapping['systems'][system][alarm.alarm_state] = 1 if system == sys else 0

print(f'\nMapping of priority pr. newState at {field}:')
mat = []
for entry in state_mapping['priorities'].values():
    mat.append([])
    for count in entry.values():
        mat[-1].append(count)
col_headers = [priorities[key] if key in priorities else f"'{key}'" for key in state_mapping['priorities']['1']]
row_headers = [key for key in state_mapping['priorities']]
print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n')

print(f'Mapping of newState pr. system at {field}:')
mat = []
for entry in state_mapping['systems'].values():
    mat.append([])
    for count in entry.values():
        mat[-1].append(count)
col_headers = [key for key in state_mapping['systems']['01']]
row_headers = [systems[field][key] if key in systems[field] else f"'{key}'" for key in state_mapping['systems']]
print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n')

print(f'Mapping of priority pr. alarmState at {field}:')
mat = []
for entry in alarm_mapping['priorities'].values():
    mat.append([])
    for count in entry.values():
        mat[-1].append(count)
col_headers = [priorities[key] if key in priorities else f"'{key}'" for key in alarm_mapping['priorities'][None]]
row_headers = [key for key in alarm_mapping['priorities']]
print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n')

print(f'Mapping of alarmState pr. system at {field}:')
mat = []
for entry in alarm_mapping['systems'].values():
    mat.append([])
    for count in entry.values():
        mat[-1].append(count)
```

```
    col_headers = [key for key in alarm_mapping['systems']['01']]
    row_headers = [systems[field][key] if key in systems[field] else f"'{key}'" for key in alarm_mapping['systems']]
    print(f'{pd.DataFrame(mat, columns=col_headers, index=row_headers)}\n\n')
```

Mapping of priority pr. newState at ULA:

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'5'
1	135	148	444	290	0	231
3	152	221	1097	231	0	241
5	151	135	1053	110	0	116
7	18	126	660	550	0	234

Mapping of newState pr. system at ULA:

	1	3	5	7
ESD	0	0	0	0
FnG	225	507	491	283
PSD	0	0	0	0
PCS	459	547	342	760
AC400	330	641	614	310
'00'	234	247	118	235

Mapping of priority pr. alarmState at ULA:

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'5'
NaN	0	1	1	10	0	3
ACT	170	328	1723	773	0	472
ABL	0	37	64	52	0	72
RTN	286	240	1466	346	0	275
SLV	0	24	0	0	0	0

Mapping of alarmState pr. system at ULA:

	NaN	ACT	ABL	RTN	SLV
ESD	0	0	0	0	0
FnG	0	790	1	715	0
PSD	0	0	0	0	0
PCS	12	1282	80	710	24
AC400	0	915	72	908	0
'00'	3	479	72	280	0

Mapping of priority pr. newState at VAL:

	Pri.1	Pri.2	Pri.3	Pri.4	EVENT	'5'
1	36	15	340	252	0	500
3	36	34	550	150	0	18
5	38	27	489	94	0	1
7	40	21	525	8	0	0

```
Mapping of newState pr. system at VAL:
         1    3    5    7
ESD      0    3    4    5
FnG      0    2    2    2
PSD     84   63   32  112
PCS    251  393  364  319
HVAC     0    0    0    0
SCMS   183  148  103   69
AC400   69  152  138   87
'00'   554   25    5    0
'50'     2    2    1    0

Mapping of priority pr. alarmState at VAL:
     Pri.1  Pri.2  Pri.3  Pri.4  EVENT  '5'
NaN     33      0    348     71      0  291
RTN     47     42    636    297      0  217
ACT     70     53    880    136      0   11
ABL      0      2     40      0      0    0

Mapping of alarmState pr. system at VAL:
       NaN  RTN  ACT  ABL
ESD      0    4    8    0
FnG      0    2    4    0
PSD      0  116  175    0
PCS    309  467  551    0
HVAC     0    0    0    0
SCMS    86  231  186    0
AC400    3  192  209   42
'00'   345  224   15    0
'50'     0    3    2    0
```

The following cell plots all alarms with a `severity` above `min_severity`, a `newState` equal to either value in `new_state_criterias` and an `alarmState` equal to either value in `alarm_state_criterias`. The plot shows number of alarms pr. shift.
E.g. `min_severity = 800` shows all Priority 1 and 2 alarms. `900` shows only Priority 1.

In [32]:

```
#%matplotlib widget
desired_alarms = {}
for field, data in alarm_states.items():
    min_severity = 800
    new_state_criterias = ['3']
    alarm_state_criterias = [None, 'ACT', 'SLV']
    temp_bins = {}
    for severity, bins in data['severities'].items():
        if int(severity) > min_severity and int(severity)%50 != 0:
            pri = severity[0]
            if pri in temp_bins:
                while len(temp_bins[pri]) < len(bins):
                    temp_bins[pri].append([])
            for i, b in enumerate(bins):
                for alarm in b:
                    if alarm.new_state in new_state_criterias and alarm.alarm_state in alarm_state_criterias:
                        if pri not in temp_bins: temp_bins[pri] = [[] for _ in range(len(bins))]
                        temp_bins[pri][i].append(alarm)
    if temp_bins:
        desired_alarms[field] = temp_bins
        fig, ax = plt.subplots()
        total = 0
        occasions = 0
        bottom = []
        for pri, bins in temp_bins.items():
            while len(bottom) < len(bins): bottom.append(0)
            ax.bar(list(range(0, -len(bins), -1)), [len(b) for b in bins], bottom=bottom[:len(bins)], label=priorities[pri])
            for i, b in enumerate(bins):
                if bottom[i] == 0 and len(b) > 0:
                    occasions += 1
                bottom[i] += len(b)
                total += len(b)
        ax.axhline(total/occasions, color='g', linestyle=':')
        ax.axhline(total/len(bottom), color='r', linestyle=':')
        ax.set_title(f'Alarms with severity >{min_severity}, newState among {new_state_criterias} and alarmState among {alarm_stat
e_criterias} at {handover.fields[field].name} pr. shift (ave. {total/occasions:.1f} pr. occasion, {total/len(bottom):.1f} pr. shif
t, {total} total)')
        ax.legend(loc='upper right', shadow=False)
        fig.set_figwidth(10)
```

```
        plt.show()
    else:
        print(f'Found no alarms matching these criterias at {field}:')
        print(f'severity: >{min_severity}')
        print(f'newState: {new_state_criterias}')
        print(f'alarmState: {alarm_state_criterias}')
```

In [33]:

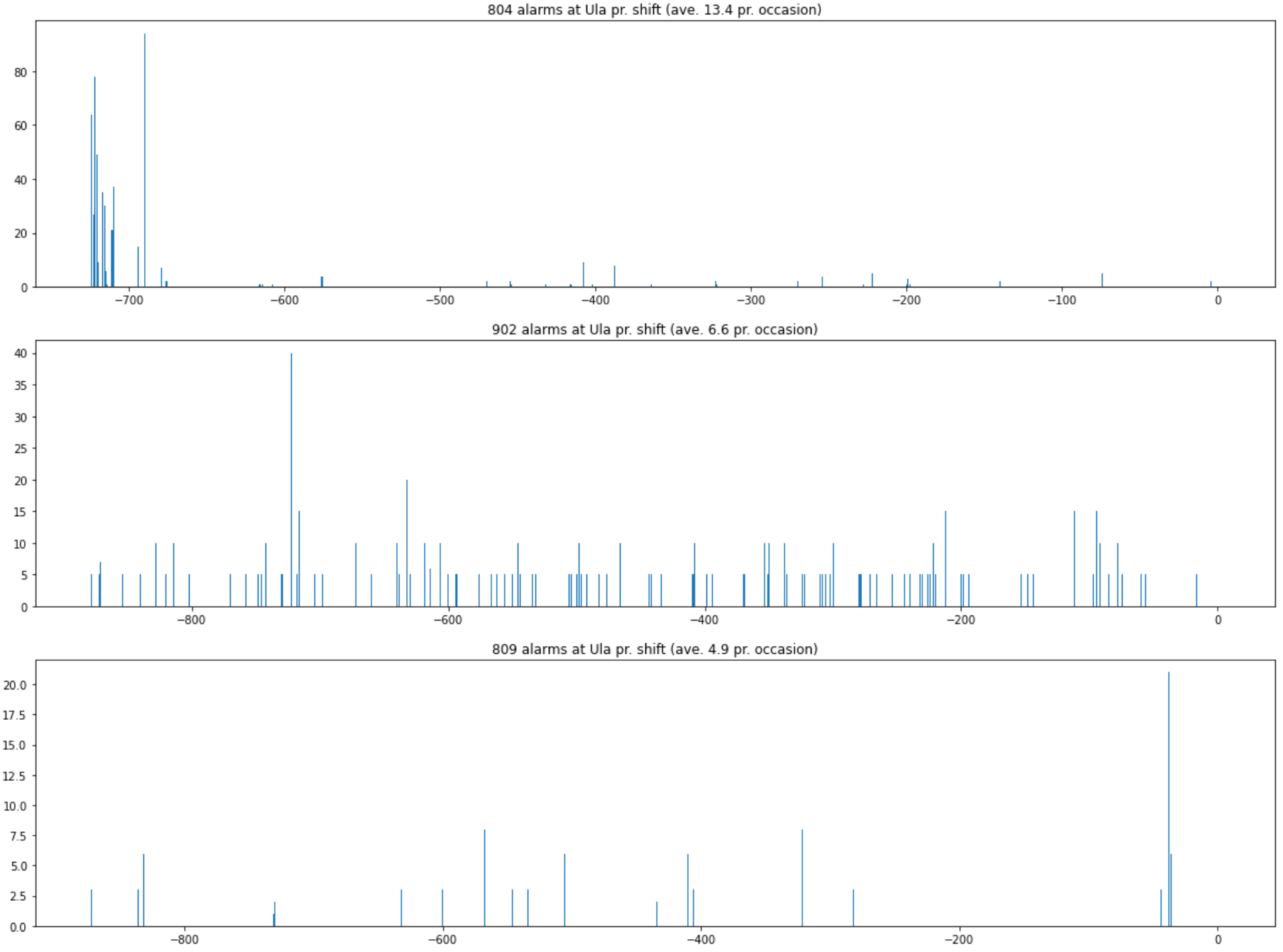
```
for field, data in desired_alarms.items():
    n_shifts = 10
    print(f'\n\nAlarms from {field} the last {n_shifts} shifts:\n')
    for shift_index in range(n_shifts):
        temp_alarms = {}
        for pri, bins in data.items():
            if bins[shift_index]: temp_alarms[pri] = bins[shift_index]
        if temp_alarms:
            print(f'\n  Alarms from {shift_index} shifts ago:')
            for pri, alarms in temp_alarms.items():
                #print(f'\n    Priority {pri}:')
                events = client.events.retrieve_multiple(ids=[alarm.id for alarm in alarms])
                for event in events:
                    #print(f'      {ms_to_datetime(event.created_time).strftime("%Y-%m-%d %H:%M:%S")} ({event.metadata["severit
y"]}) {event.metadata["source"]}: {event.metadata["ObjectDescription"]}, {event.description}')
                    severity = event.metadata["severity"]
                    pri = priorities[severity[0]]
                    sys = systems[field][severity[1:]]
                    #print(f'      {event.metadata["time"]} | {event.metadata["source"]} | {event.metadata["ObjectDescription"]} |
{event.description} ({pri} {sys})')
                    print_event(event)
                    #print(event)
                    asset = client.assets.retrieve(id=event.asset_ids[0])
                    #print(asset)
```

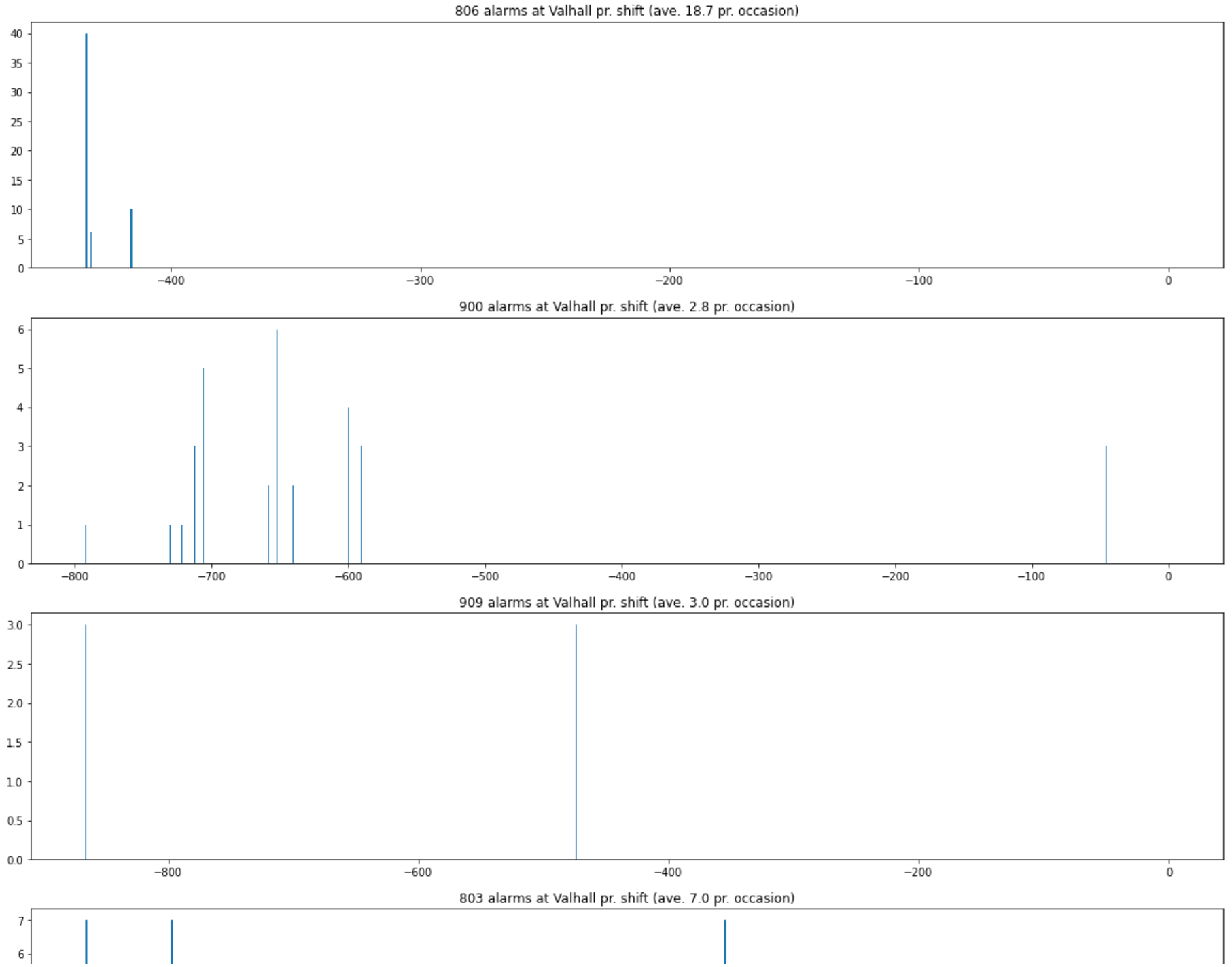
```
Alarms from ULA the last 10 shifts:



Alarms from VAL the last 10 shifts:
```

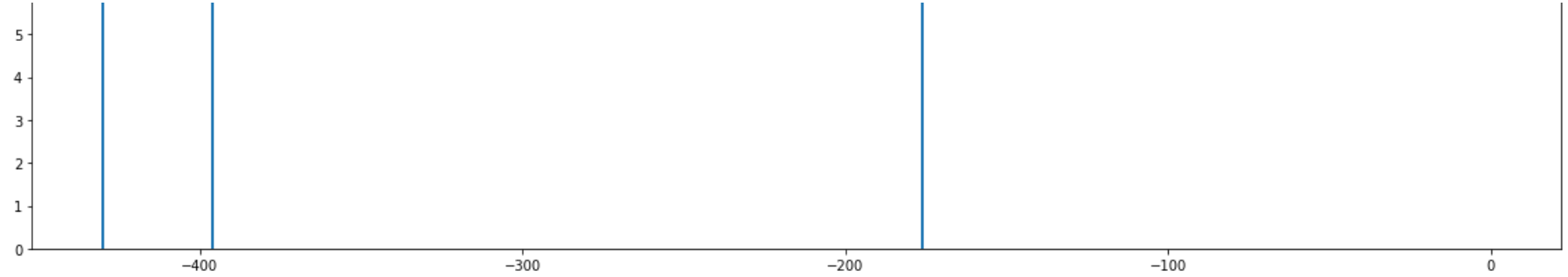
The following cell plots all alarms with a severity above `min_severity` and a `newState` equal to `wanted_state`. Each plot shows number of alarms pr. shift. E.g. `min_severity = 800` shows all Priority 1 and 2 alarms. `900` shows only Priority 1.

In [34]:

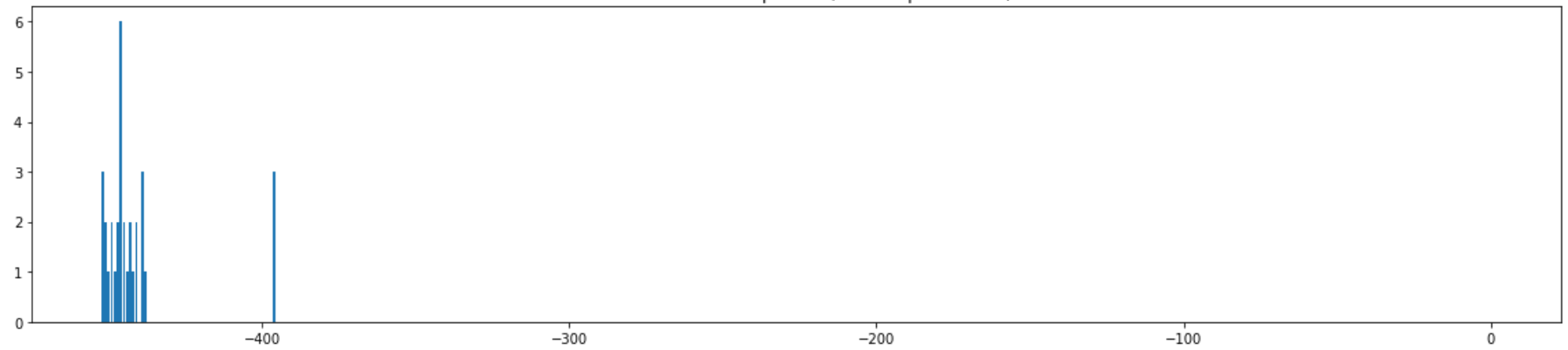
```
%matplotlib inline
for field, severities in severity_mapping.items():
    fig, axs = plt.subplots(len([sev for sev in severities if int(sev) > 800]), 1, sharex=False)
    fig_index = 0
    for severity, bins in severities.items():
        if int(severity[0]) >= 8:
            total = 0
            occasions = 0
            for b in bins:
                if b > 0:
                    total += b
                    occasions += 1
            axs[fig_index].bar(list(range(0, -len(bins), -1)), bins)
            axs[fig_index].set_title(f'{severity} alarms at {handover.fields[field].name} pr. shift (ave. {total/occasions:.1f} p
r. occasion)')
            fig_index += 1
    fig.set_figheight(fig_index*5)
    fig.set_figwidth(20)
    plt.show()
```


804 alarms at Ula pr. shift (ave. 13.4 pr. occasion)
902 alarms at Ula pr. shift (ave. 6.6 pr. occasion)
809 alarms at Ula pr. shift (ave. 4.9 pr. occasion)

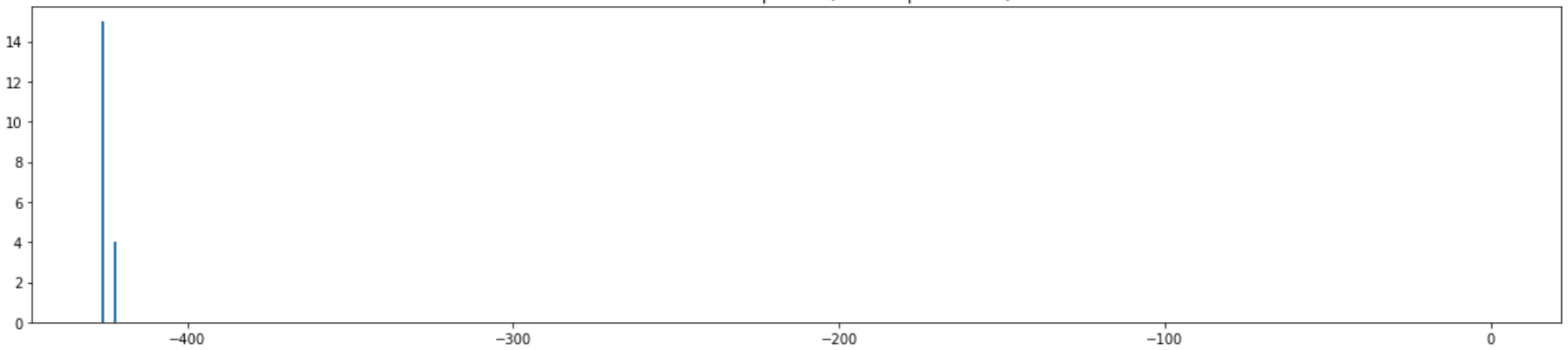

806 alarms at Valhall pr. shift (ave. 18.7 pr. occasion)
40
35
30
25
20
15
10
5
0
−400
−300
−200
−100
0
900 alarms at Valhall pr. shift (ave. 2.8 pr. occasion)
6
5
4
3
2
1
0
−800
−700
−600
−500
−400
−300
−200
−100
0
909 alarms at Valhall pr. shift (ave. 3.0 pr. occasion)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−800
−600
−400
−200
0
803 alarms at Valhall pr. shift (ave. 7.0 pr. occasion)
7
6



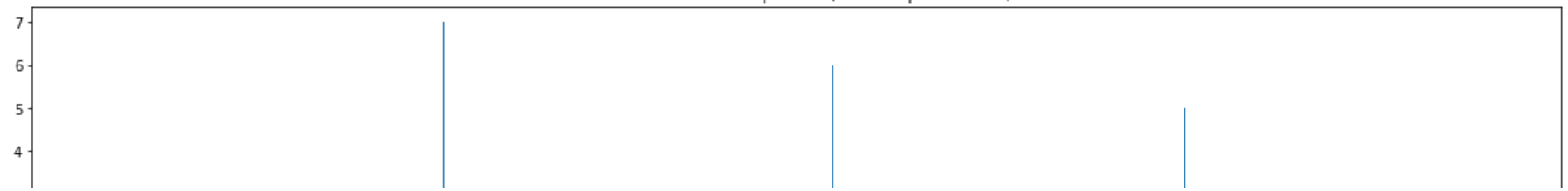
809 alarms at Valhall pr. shift (ave. 2.1 pr. occasion)

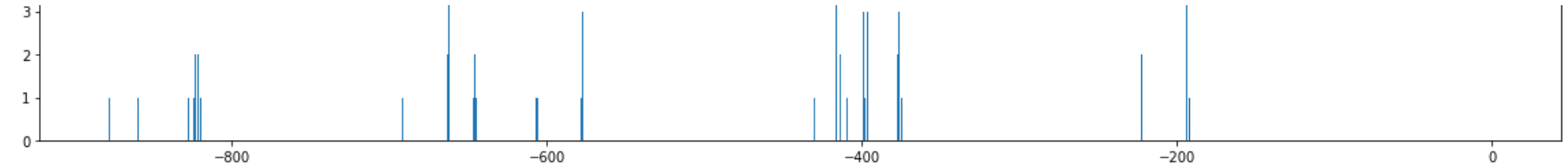


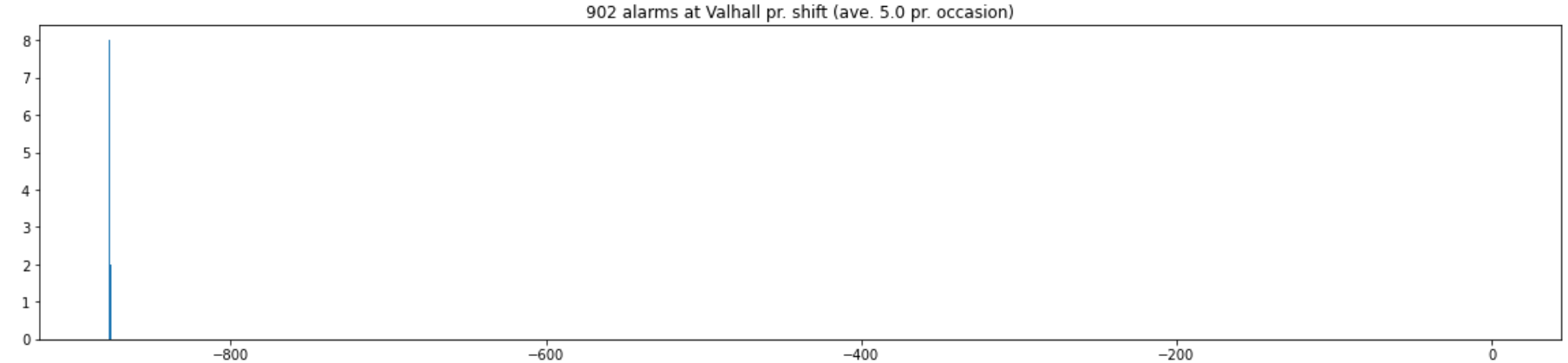
901 alarms at Valhall pr. shift (ave. 9.5 pr. occasion)

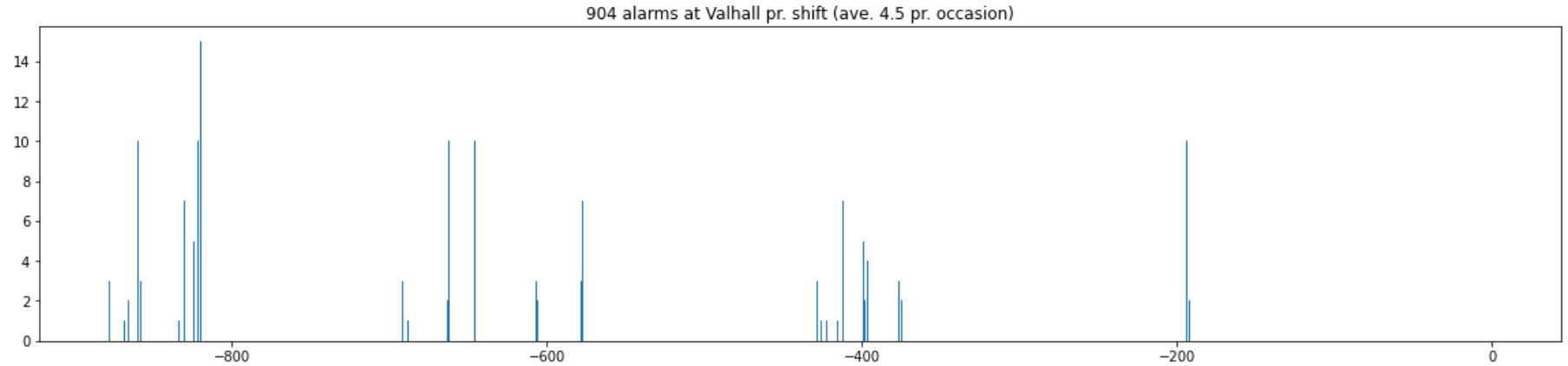


802 alarms at Valhall pr. shift (ave. 1.9 pr. occasion)




3
2
1
0
−800
−600
−400
−200
0


902 alarms at Valhall pr. shift (ave. 5.0 pr. occasion)
8
7
6
5
4
3
2
1
0
−800
−600
−400
−200
0

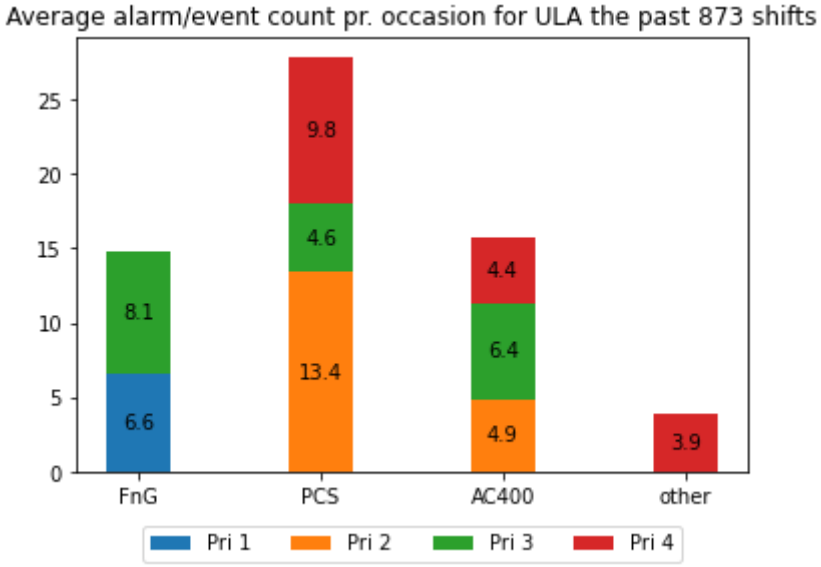

904 alarms at Valhall pr. shift (ave. 4.5 pr. occasion)
14
12
10
8
6
4
2
0
−800
−600
−400
−200
0

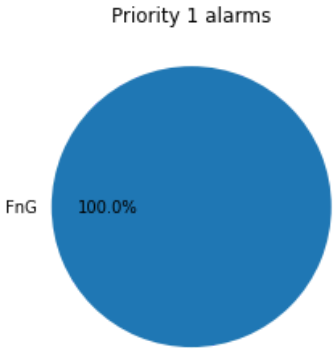
In [ ]:

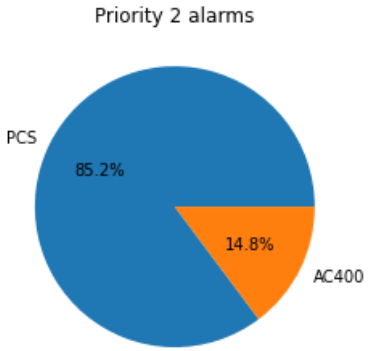
In [35]:

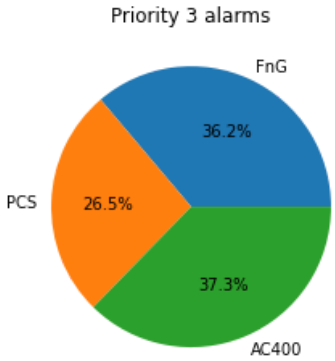
```
for field, severities in severity_mapping.items():
    alarm_priorities = {
        '9': 'pri1',
        '8': 'pri2',
        '7': 'pri3',
        '6': 'pri4',
        '4': 'event',
        'other': 'other'
    }
    alarm_systems = {
        '01': 'ESD',
        '02': 'FnG',
        '03': 'PSD',
        '04': 'PCS',
        '05': 'HVAC',
        '06': 'SCMS',
        '09': 'AC400',
        'other': 'other'
    }
    mapping = {pri: {sys: 0 for sys in alarm_systems.values()} for pri in alarm_priorities.values()}
    occasions = {pri: {sys: 0 for sys in alarm_systems.values()} for pri in alarm_priorities.values()}

    for severity, bins in severities.items():
        pri = alarm_priorities[severity[0]] if severity[0] in alarm_priorities else 'other'
        sys = alarm_systems[severity[1:]] if severity[1:] in alarm_systems else 'other'

        n_shifts = None
        temp_bins = bins[0:min(len(bins), n_shifts) if n_shifts else len(bins)]
        n_shifts = max(n_shifts, len(temp_bins)) if n_shifts else len(temp_bins)
        for b in temp_bins:
            if b > 0:
                mapping[pri][sys] += b
                occasions[pri][sys] += 1

    matrix = []
    row_headers = [pri for pri in mapping.keys()]
    col_headers = None
    for pri, entries in mapping.items():
        row = []
```

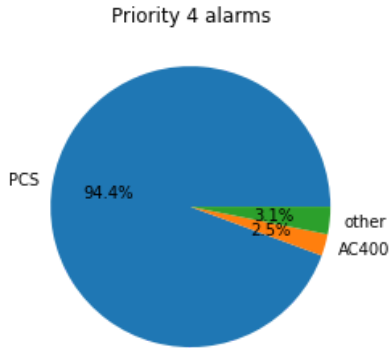
```
    if not col_headers: col_headers = [sys for sys in entries.keys()]
    for count in entries.values():
        row.append(count)
    matrix.append(row)

occasion_matrix = []
for pri, entries in occasions.items():
    row = []
    if not col_headers: col_headers = [sys for sys in entries.keys()]
    for count in entries.values():
        row.append(count)
    occasion_matrix.append(row)

width = 0.35
bottom = [0 for i in range(len(matrix[0]))]
for i in range(4):
    labels = []
    values = []
    temp = []
    for j in range(len(matrix[i])):
        if matrix[i][j] > 0:
            labels.append(col_headers[j])
            values.append(matrix[i][j]/occasion_matrix[i][j])
            temp.append(bottom[j])
            bottom[j] += matrix[i][j]/occasion_matrix[i][j]
    p = plt.bar(labels, values, width, bottom=temp, label=f'Pri {i+1}')
    plt.bar_label(p, fmt='%.1f', label_type='center')
plt.title(f'Average alarm/event count pr. occasion for {field} the past {n_shifts} shifts')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
      ncol=4, shadow=False)
plt.show()

fig, axs = plt.subplots(1, 4)
for i, row in enumerate(matrix[0:4]):
    sizes = [num for num in row if num != 0]
    labels = [col_headers[i] for i in range(len(row)) if row[i] != 0]
    axs[i].pie(sizes, labels=labels, autopct='%.1f%%')
    axs[i].set_title(f'Priority {i+1} alarms')
fig.set_figwidth(20)
plt.show()
```


Average alarm/event count pr. occasion for ULA the past 873 shifts
25
20
15
10
5
0
9.8
4.6
8.1
13.4
4.4
6.4
6.6
4.9
3.9
FnG
PCS
AC400
other
Pri 1
Pri 2
Pri 3
Pri 4

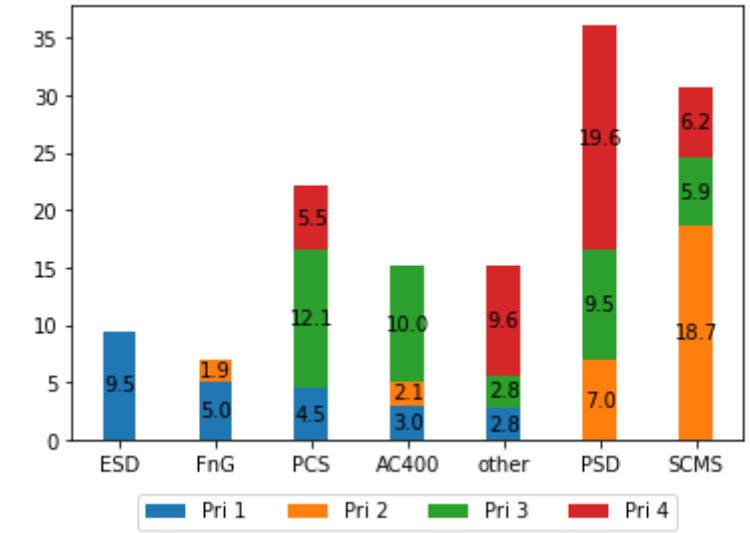

Priority 1 alarms
FnG
100.0%

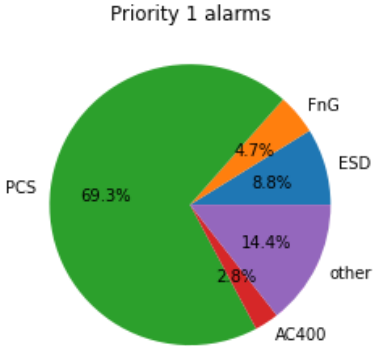

Priority 2 alarms
PCS
85.2%
14.8%
AC400

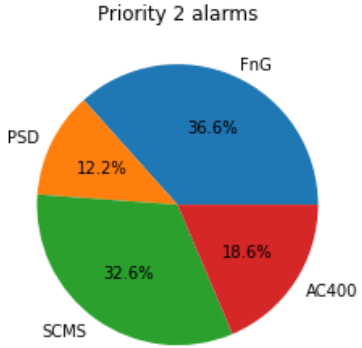

Priority 3 alarms
FnG
36.2%
PCS
26.5%
37.3%
AC400

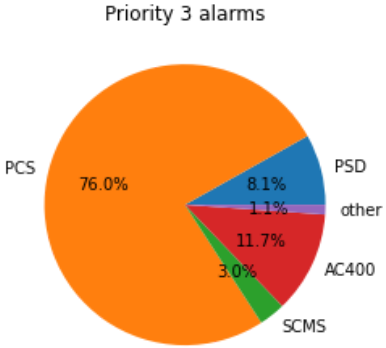

Priority 4 alarms
PCS
94.4%
3.1%
other
2.5%
AC400

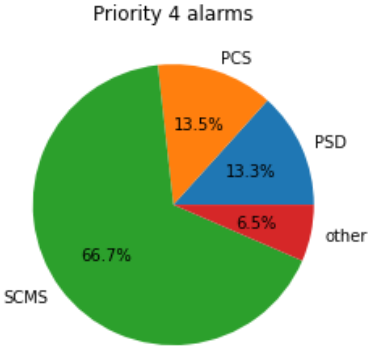
Average alarm/event count pr. occasion for VAL the past 879 shifts




Priority 1 alarms
FnG
4.7%
ESD
8.8%
PCS
69.3%
14.4%
other
2.8%
AC400


Priority 2 alarms
FnG
36.6%
PSD
12.2%
18.6%
AC400
32.6%
SCMS


Priority 3 alarms
PCS
76.0%
PSD
8.1%
other
1.1%
11.7%
AC400
3.0%
SCMS


Priority 4 alarms
PCS
13.5%
PSD
13.3%
6.5%
other
66.7%
SCMS

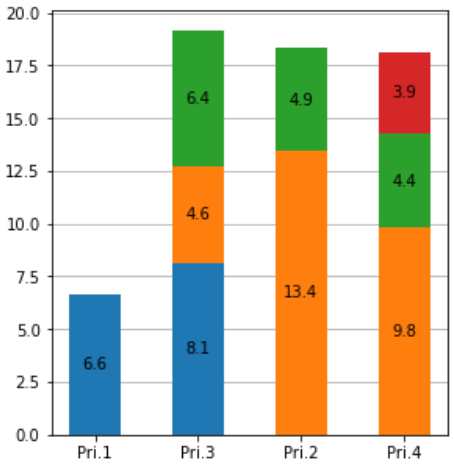
In [36]:

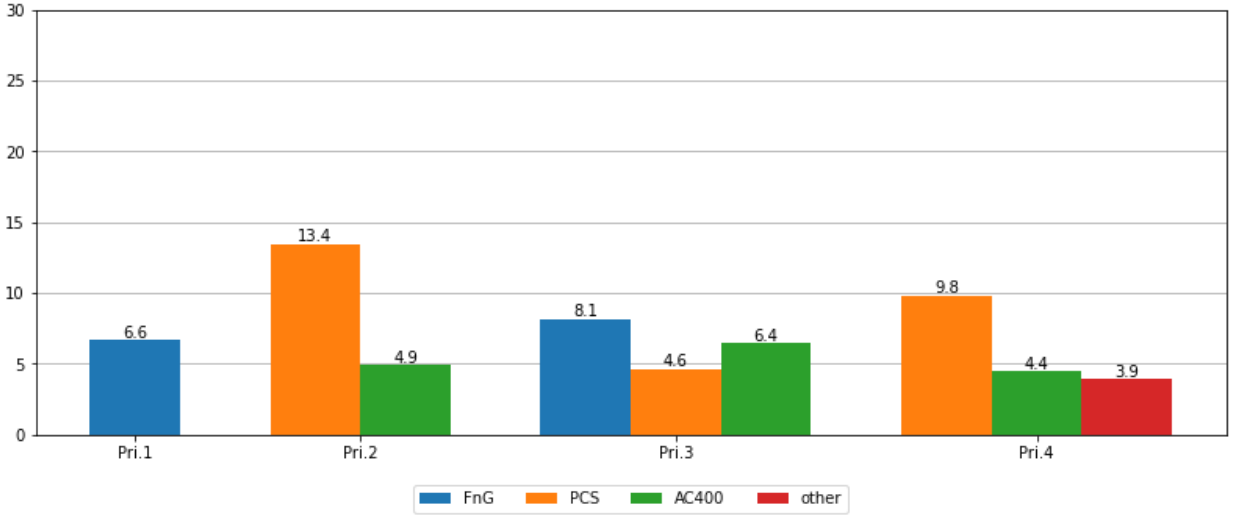
```
for field, severities in severity_mapping.items():
    severity_dict = {sys: {pri: {'total': 0, 'occasions': 0, 'average': 0} for pri in priorities} for sys in systems[field]}

    for severity, bins in severities.items():
        n_events = 0
        count = 0
        for b in bins:
            if b > 0:
                n_events += b
                count += 1
        sys = severity[1:]
        pri = severity[0]
        if sys not in severity_dict: severity_dict[sys] = {priority: {'total': 0, 'occasions': 0, 'average': 0} for priority in se
verity_dict['01']}
        if pri in severity_dict[sys]:
            severity_dict[sys][pri]['total'] = n_events
            severity_dict[sys][pri]['count'] = count
            severity_dict[sys][pri]['average'] = n_events/count
        else:
            for system in severity_dict:
                severity_dict[system][pri] = {'total': n_events, 'occasions': count, 'average': n_events/count} if system == sys e
lse {'total': 0, 'occasions': 0, 'average': 0}
    mat = []
    for entry in severity_dict.values():
        mat.append([])
        for data in entry.values():
            mat[-1].append(data['average'])
    col_headers = [priorities[key] if key in priorities else f"'{key}'" for key in severity_dict['01']]
    row_headers = [systems[field][key] if key in systems[field] else f"'{key}'" for key in severity_dict]

    fig, axs = plt.subplots(1, 2, figsize=(20,5), gridspec_kw={'width_ratios': [1, 3]})
    width = 0.5
    bottom = [0 for i in range(4)]
    for sys, entries in severity_dict.items():
        labels = []
        values = []
        temp = []
        max_value = 1000
        for i, pri in enumerate(entries):
```

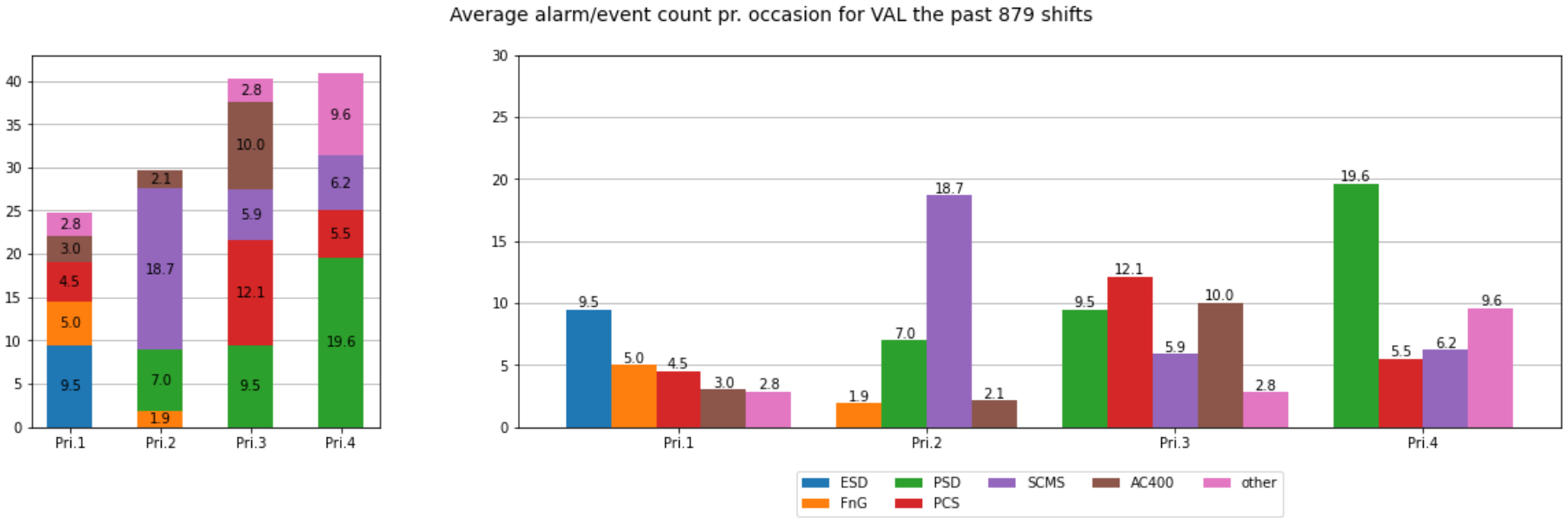
```
            if entries[pri]['average'] != 0:
                value = entries[pri]['average']
                if value > max_value:
                    max_value = value
                else:
                    labels.append(priorities[pri])
                    values.append(value)
                    temp.append(bottom[i])
                    bottom[i] += value
            if i >= 3: break
        if labels:
            p = axs[0].bar(labels, values, width, bottom=temp, label=f'{systems[field][sys] if sys in systems[field] else "other"}
', zorder=3)
            axs[0].bar_label(p, fmt='%.1f', label_type='center')
    axs[0].yaxis.grid(zorder=1)
    #axs[0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=4, shadow=False)

    width = 0.35
    labels = [pri for pri in priorities.values()]
    labels = labels[:-1]
    n_labels = [0 for _ in range(len(labels))]
    legends = []
    values = []
    for sys, entries in severity_dict.items():
        legends.append(systems[field][sys] if sys in systems[field] else 'other')
        values.append([])
        for i, pri in enumerate(entries):
            if i >= len(labels): break
            values[-1].append(entries[pri]['average'])
            if values[-1][i] != 0:
                n_labels[i] += 1
    x_values = []
    x_centres = [0]
    for i in range(len(n_labels)-1):
        x_centres.append(x_centres[i] + (n_labels[i] + n_labels[i+1] + 2)*width/2)
    fig_width = 0
    for i, n in enumerate(n_labels):
        x_values.append(np.linspace(x_centres[i] + (1-n)*width/2, x_centres[i] + (n-1)*width/2, n))
        fig_width += n
    #fig.set_figwidth(fig_width)
    for i, row in enumerate(values):
        x = []
```

```
        y = []
        for j in range(len(row)):
            if row[j] != 0:
                x.append(x_values[j][0])
                x_values[j] = x_values[j][1:]
                y.append(row[j])
        if x:
            p = axs[1].bar(x, y, width, label=legends[i], zorder=3)
            axs[1].bar_label(p, fmt='%.1f')
    axs[1].set_xticks(x_centres, labels)
    axs[1].set_ylim(0, 30)
    axs[1].legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=5, shadow=False)
    axs[1].yaxis.grid(zorder=1)

    fig.suptitle(f'Average alarm/event count pr. occasion for {field} the past {n_shifts} shifts', fontsize=14)
    plt.show()
```

Average alarm/event count pr. occasion for ULA the past 879 shifts







Check if assets outside of the OPC UA dataset has OPC UA events

Hardcoded section to find the rest of the assets outside of the OPC UA datasets, so that we can check if they can have events from the OPC UA datasets related to them, even though the source of the asset (typically) is aveva:

In [37]:

```
possible_assets = {}
for field in handover.fields.values():
    possible_assets[field.abbr()] = []
    n_comp = 0
    count = 0
    event_count = 0
    for comp in field.main_components:
        #if not comp.assets:
        if not comp.exact_match:
            n_comp += 1
            assets = client.assets.list(name=comp.name)
            if assets:
                count += 1
                asset_list = []
                for asset in assets:
                    if re.search(f'^{comp.name}', asset.name):
                        if field.abbr() == 'VAL':
                            if asset.external_id[0:3] in ['VAL', 'VLA', 'VFS', 'VFN', 'VFW', 'TAM', 'HOP', 'HOD']:
                                asset_list.append(asset)
                        elif field.abbr() == 'ULA':
                            if asset.external_id[0:3] in ['ULA', 'TAM']:
                                asset_list.append(asset)
                        else:
                            if field.abbr() == asset.external_id[0:3]:
                                asset_list.append(asset)
                if len(asset_list) > 1:
                    print('found multiple assets')
                    for asset in asset_list:
                        print(asset.external_id)
                        if not re.search(f'^{comp.name}', asset.name):
                            asset_list.remove(asset)
                if asset_list:
                    possible_assets[field.abbr()].append(asset_list[0])
                    if handover.fields[field.abbr()].data_set_id:
                        datasets = [handover.fields[field.abbr()].data_set_id]
                        #datasets.extend([1525574569706251, 3874712032478167])
                        event_count += len(asset_list[0].events(data_set_ids=datasets, limit=-1))
            #else:
            #    print(f'Could not find any asset with the name {comp.name} on {field.name}')
```

```
    print(f'Found {count} out of {n_comp} of the other main component tags outside the OPC UA datasets on {field.name}')
    print(f'These tags had {event_count} OPC UA events related to them')
```

```
Found 232 out of 233 of the other main component tags outside the OPC UA datasets on Alvheim
These tags had 0 OPC UA events related to them
Found 71 out of 71 of the other main component tags outside the OPC UA datasets on Ivar Aasen
These tags had 0 OPC UA events related to them
Found 501 out of 517 of the other main component tags outside the OPC UA datasets on Skarv
These tags had 0 OPC UA events related to them
Found 21 out of 21 of the other main component tags outside the OPC UA datasets on Ula
These tags had 0 OPC UA events related to them
Found 54 out of 54 of the other main component tags outside the OPC UA datasets on Valhall
These tags had 0 OPC UA events related to them
```

Looping through shifts in reverse chronological order until it finds a shift where there are logged events on the relevant assets:

In [39]:

```
today = datetime.datetime.today()
date = today - datetime.timedelta(days=19)    # Start search before TAR the 25th/26th (Ula/Valhall) of May
t2 = 0
t1 = 0

for field, comps in handover.main_components.items():
    global t1, t2
    t2 = datetime.datetime(date.year, date.month, date.day, 19)
    t1 = t2 - datetime.timedelta(hours=12)
    found = False
    shifts_ago = 0
    n_events = 0
    while not found:
        print(t1)
        for comp in comps:
            for asset in comp.assets:
                temp = asset.parent.events(data_set_ids=[handover.fields[field].data_set_id], last_updated_time={"min": timestamp_
to_ms(t1), "max": timestamp_to_ms(t2)}, sort=['lastUpdatedTime:asc'], limit=-1)
                if temp:
                    found = True
                    asset.events = temp
                    n_events += len(temp)
        if not found:
            t1 -= datetime.timedelta(hours=12)
            t2 -= datetime.timedelta(hours=12)
            shifts_ago += 1
    print(f'Found {n_events} events at {handover.fields[field].name} {shifts_ago} shifts ago, between {t1} and {t2}.')
    #break
```

```
2022-12-21 07:00:00
Found 30 events at Ula 0 shifts ago, between 2022-12-21 07:00:00 and 2022-12-21 19:00:00.
2022-12-21 07:00:00
Found 41 events at Valhall 0 shifts ago, between 2022-12-21 07:00:00 and 2022-12-21 19:00:00.
```

In [40]:

```
handover.main_components['ULA'][7].assets[0].parent.events(data_set_ids=[handover.fields['ULA'].data_set_id], last_updated_time={
"min": timestamp_to_ms(t1), "max": timestamp_to_ms(t2)}, sort=['lastUpdatedTime:asc'], limit=-1)
```

Out[40]:

In [42]:

```
for field, comps in handover.main_components.items():
    print(f'\n\nEvents from {handover.fields[field].name}:')
    severity_count = [0, 0, 0, 0, 0, 0]
    for comp in comps:
        for asset in comp.assets:
            if asset.events:
                for event in asset.events:
                    severity = event.metadata['severity']
                    if "90" in severity:
                        print(severity)
                        severity_count[0] += 1
                    elif "80" in severity:
                        severity_count[1] += 1
                    elif "70" in severity:
                        severity_count[2] += 1
                    elif "60" in severity:
                        severity_count[3] += 1
                    elif "40" in severity:
                        severity_count[4] += 1
                    else:
                        severity_count[5] += 1
    print(f'Priority 1: {severity_count[0]}\nPriority 2: {severity_count[1]}\nPriority 3: {severity_count[2]}\nPriority 4: {severi
ty_count[3]}\nEvents: {severity_count[4]}\nOther: {severity_count[5]}')
```

```
Events from Ula:
Priority 1: 0
Priority 2: 21
Priority 3: 0
Priority 4: 0
Events: 0
Other: 9


Events from Valhall:
Priority 1: 0
Priority 2: 0
Priority 3: 0
Priority 4: 0
Events: 40
Other: 1
```

In [43]:

```
for field, comps in handover.main_components.items():
    print(f'\n\nEvents from {handover.fields[field].name}:')
    for comp in comps:
        comp_title = False
        for asset in comp.assets:
            if asset.events:
                if not comp_title:
                    print(f'\n\n{comp.name} ({comp.description}):')
                    comp_title = True
                print(f'\n{asset.name} ({len(asset.events)} events):')
                for event in asset.events:
                    print(f'{ms_to_datetime(event.last_updated_time).strftime("%Y-%m-%d %H:%M:%S")} ({event.metadata["severity"]})
{event.description}')
```

Events from Ula:

P-0425 (WI Pump A):

PM-0425 (2 events):
2022-12-21 07:44:03 (200) Acknowledge of alarms on object S-1003-11_PM-0425 has been requested.
2022-12-21 07:44:17 (500) Loss of Control Voltage

P-02101 (Seawater lift pump):

P-02101-A-33 (28 events):
2022-12-21 10:45:32 (809)   91.00 %
2022-12-21 10:45:33 (809)   91.00 %
2022-12-21 10:45:33 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 10:45:35 (809)   91.00 %
2022-12-21 10:45:36 (809)   91.00 %
2022-12-21 10:45:38 (809)   91.00 %
2022-12-21 10:45:39 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 10:45:40 (809)   91.00 %
2022-12-21 13:46:19 (809)   91.00 %
2022-12-21 13:46:23 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 13:46:23 (809)   91.00 %
2022-12-21 13:46:25 (809)   91.00 %
2022-12-21 13:46:28 (809)   91.00 %
2022-12-21 13:46:30 (809)   91.00 %
2022-12-21 13:46:30 (809)   91.00 %
2022-12-21 13:46:32 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 13:46:32 (809)   91.00 %
2022-12-21 13:46:32 (809)   91.00 %
2022-12-21 13:46:33 (809)   91.00 %
2022-12-21 13:46:33 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 14:46:48 (809)   91.00 %
2022-12-21 14:46:48 (809)   91.00 %
2022-12-21 14:46:51 (809)   91.00 %
2022-12-21 14:46:51 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 14:46:53 (809)   91.00 %
2022-12-21 14:47:08 (200) Acknowledge of Lim H1 condition on object P-02101-A-33 has been requested.
2022-12-21 14:47:10 (809)   91.00 %

2022-12-21 14:47:13 (809)   91.00 %

Events from Valhall:

18-PA-8010 (W. Inj. Pump):

18-PA-8010-M01 (27 events):
2022-12-21 10:45:35 (409) Normal
2022-12-21 10:45:35 (409) Normal
2022-12-21 10:45:39 (409) Alarm
2022-12-21 10:45:43 (409) On
2022-12-21 10:45:43 (409) Off
2022-12-21 10:45:46 (409) Off
2022-12-21 10:45:46 (409) On
2022-12-21 10:45:46 (409) Alarm
2022-12-21 10:45:51 (409) Off
2022-12-21 10:45:51 (409) Alarm
2022-12-21 10:45:51 (200) RESET_CH_LATCH False -> True
2022-12-21 10:45:53 (409) Off
2022-12-21 10:45:57 (409) Off
2022-12-21 10:45:57 (409) Alarm
2022-12-21 10:45:57 (409) On
2022-12-21 10:46:02 (409) Normal
2022-12-21 10:46:02 (409) On
2022-12-21 10:46:02 (409) Normal
2022-12-21 11:45:50 (409) Alarm
2022-12-21 11:45:50 (409) Off
2022-12-21 11:45:54 (409) Alarm
2022-12-21 11:45:59 (409) Off
2022-12-21 11:46:01 (409) On
2022-12-21 11:46:09 (409) Normal
2022-12-21 11:46:12 (409) Normal
2022-12-21 11:46:12 (409) Normal
2022-12-21 11:46:16 (409) On

18-PA-8010-M01.H06 (14 events):
2022-12-21 10:45:39 (409) Off
2022-12-21 10:45:39 (409) On
2022-12-21 10:45:47 (409) Off
2022-12-21 10:45:50 (409) On

```
2022-12-21 10:45:50 (409) Off
2022-12-21 10:45:51 (409) On
2022-12-21 10:45:57 (409) On
2022-12-21 10:46:02 (409) Off
2022-12-21 11:46:03 (409) On
2022-12-21 11:46:07 (409) Off
2022-12-21 11:46:09 (409) On
2022-12-21 11:46:09 (409) Off
2022-12-21 11:46:12 (409) Off
2022-12-21 11:46:16 (409) On
```

In [44]:

```
print(handover.assets['ULA'][0])
print(events['ULA']['P-01102-B'][0])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-44-78152be48b06> in <module>
----> 1 print(handover.assets['ULA'][0])
      2 print(events['ULA']['P-01102-B'][0])

AttributeError: 'Project' object has no attribute 'assets'
```