Feature Specification for

SiESoCom

Raw Acoustic Filer 'RAF'

1.1

Revision History

Version	Who	When	Content	Comment
1.0	EP	May 17 2021	Baseline	From former section 3 of F101 spec Some mods including topic names, run_id addition in model
1.1	EP	July 2 2021	Update	Update Acquisition logic requirement

Table of Content

1 Context & Solution Topology

This document outlines the architecture and system requirements for the SiESoCom filer application

1.1 System topology views

The SiESoCom application is a multi-tier application which topology is depicted below. This specification deals with the raw acoustic filer application (6)

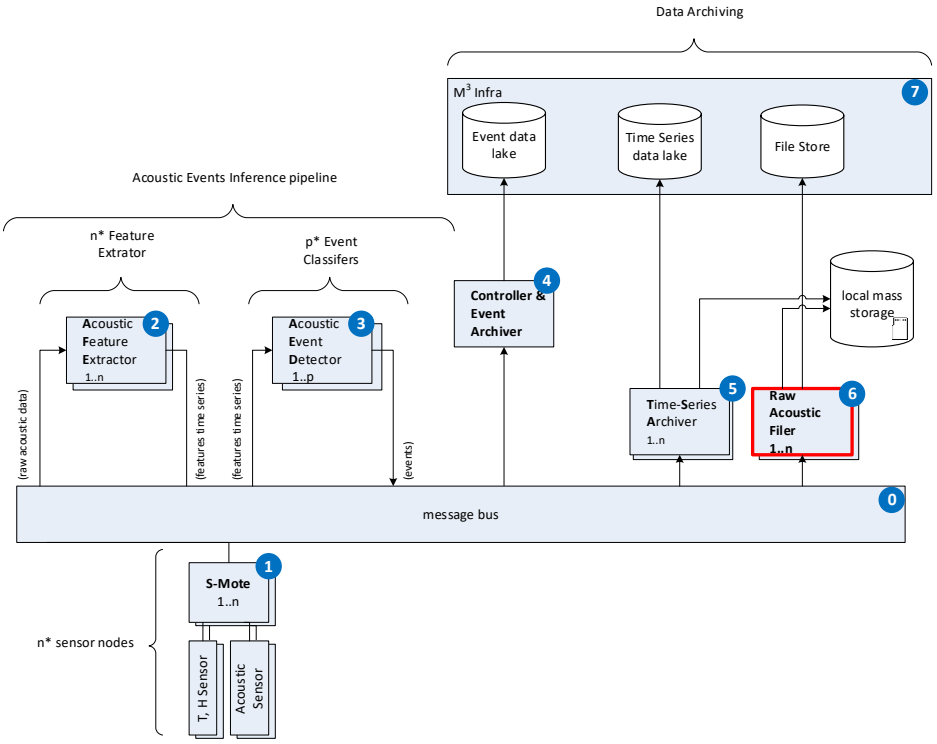

Fig: SiESoCom application topology – all tiers –

2 Filer Application

This section details the specifications for a M³ raw audio filer application, leveraging the capabilities described on the "F101 Local File Storage" specifications

2.1 Assumptions

We are assuming here that acoustic sensors (e.g. S-motes) are publishing raw audio to a M³ managed MQTT broker. The raw audio frames structure is defined in the "Data Schema" config parameter of the S-mote sensors. At this time we support only raw audio frame without header of the following sizes: 1024, 2048, 4096 or 8192 bytes.

If the operator has not inserted a mass storage device before requesting the acquisition to start, the app shall show some error/warning message.

Raw audio data is streaming in mqtt broker on one or multiple topics built per the following template:

<setup name>device/acoustic/<device S/N>/<sensor S/N>

e.g.

SiESoComSetup_1/device/acoustic/CSP-B827EB2DB0F1/001

The Filer application is expected to produce the Audio files as uncompressed .WAV files

2.2 User experience elements

(a) User commissions and starts an acoustic monitoring setup. The setup can include multiple sensor devices and each device can have multiple acoustic sensors. Each sensor source endpoint is configured individually in terms of {frame_size, sampling_frequency, sample_size, channel count, mic_gain}.
(b) User enables/disables the acoustic streaming endpoints.
(c) User commissions and starts one of more Filer application(s).
(d) User configures each Filer application:
    - Choose the source endpoint(s) from all possible endpoints
    - Set Acquisition run details:
        - 'tag'
        - 'frames_per_file'
        - 'acq_duration' (min)
    - Set the acquisition storage: cloud or local. (The cloud case is not discussed here but should be supported anyway)
(e) User inserts the USB mass storage device if not already done.
(f) User request the acquisition to start immediately or at a set start_date either manually or programmatically.
(g) The acquisition shall start if all the pre-conditions are fulfilled e.g. current_date = start_date AND USB mass storage device properly mounted and available.
(h) When the acquisition actually starts, a unique Run_id is generated to keep track of the context.
(i) Acquisition proceed for the acquisition duration at the start_date:
    - Files are accumulated on the mass storage device into dedicated directories to prevent conflicts e.g. /<Setup_name>/<Tag>/<Run_Id>/*.raw

- A special 'meta-data.txt' file shall be created for each run including the details on the acquisition, including the following information:
  - Run Id,
  - Tag ,
  - Tenant Id/name,
  - Account Id/Name,
  - Setup Id/Name
  - Start date,
  - Duration,
  - Identification of the data source:
    - Device S/N,
    - Sensor S/N,
    - Sensor Configuration = {sampling_frequency, sample_size, channel count, mic_gain}
  - List of generated files with their size
  - ...

- When acquisition terminates, the app state is updated to signal that I/O are finished: This shall be used for unmount/eject logic

Notes:

The file applications belong to a given setup, so they should discover automatically the networking details of the message broker to use and how to connect to it: IP, port, TLS mode

The configuration of the selected endpoint {frame_size, sampling_frequency, sample_size,channel count, mic_gain} shall also be automatically pulled from the concerned device, probably via a custom API

2.3 Requirements & Design elements

A M3 acquisition setup can involve more that one sensor from more than one device, so either (i) the filer app is able to create files from more than one stream or (ii) we assume that we will instanciate one Filer app per stream.

All the acoustic data streams are published into the setup's mqtt message broker, on different topics, per the following pattern:

<setup name>device/acoustic/<device S/N>/<sensor S/N>

e.g.

SiESoComSetup_1/device/acoustic/CSP-B827EB2DB0F1/001

Each stream endpoint configurations are defined by the following set of config parameters:

```
{
frame_size (byte), // e.g. 8192 bytes
sampling_frequency (Hz), // e.g. 44100 Hz
```

```
sample_size (bit), // e.g 16 bits
channel (int)
mic_gain (int) // e.g 20
}
```

These attributes are set on each sensor device and shall be retrieved from there (e.g. using M^3 pub/sub or a M^3 API)

In addition to the stream endpoint information, the file creation process will require the following meta-data:

'setup_name': The unique name of the setup to which the steam endpoint belongs to.
'tag': a user-defined string meant to identify the context of the system during the acquisition e.g "turbine rampdown, run 8, day 3"
'frames_per_file': The number of frames the filer shall concatenate into each individual file. This number, together with the frame_size and the sample_size defined the file of each individual file of a given acquisition run. We should produce file of 100 to 500 MB typically, even though this is fully user defined.
Output format: The type of file to be produced. Uncompressed .raw or .wav, possibly compressed e.g MP3.

Note: During the acquisition, the state and amount of storage left on the mass storage device is reported via subscribed config parameters.

Error handling of importance :

If the amount of storage data become insufficient, the run shall be gracefully stopped and a status message shall be reported e.g "local storage full, run stopped at <TT>"

If the mass storage device is unmounted or removed, the run shall be gracefully stopped and a status message shall be reported e.g "local storage removed at <TT>"

1.1 Filer Modeling elements

Filer app should have at least the following config parameters:

Application Credentials
- CA cert
- Cert
- Key
- Expiration Date

States
- State
- Additional state information

Notice: these combinaisons of (state, additional state information) can be extended, but the application shall at least support the following 4 states: Not Started, Initializing, Ready

General
Name

Application
Data EndPoint
<list of user-selected input enpoints>

Acquisition control
Acq State (per FSM defined below)
Acq State Additional Info
Mass Storage Device State (subcribed)
Mass Storage Avail Space (subscribed)
Command (start/stop)
start_date
duration
tag
run_id

Storage Control
frames_per_file
output format = {raw, wav,...}
destination = {cloud permanent, cloud temporary, local }
expiration delay (if cloud temporary)

NOTICE: A Dedicated specification for the FILER application exist on the official doc repo:
https://docs.google.com/document/d/1Yt41i1qfQUHzcuP3d1x6EyfCZLRFGZQ5xjIEYroDZmQ/edit#heading=h.k31nb6iub5p3

The RAF application shall report it's current state and possibly additional state information at any time in the 'Acq State' and 'Acq State Additional Info' config params of the 'Acquisition Control' group. (See FSM details below)

The acquisition state transitions shall be governed by the finite state machine define hereafter.

Note: The RAF local acquisition logic shall monitor the mass storage device mounting state and remaining avalaible space. In particular:
It shall not be possible to start a local acquisition if the USB mass storage device is not mounted or out of space.
Acquisition shall gracefully stop is USB mass storage device is full or unmounted.

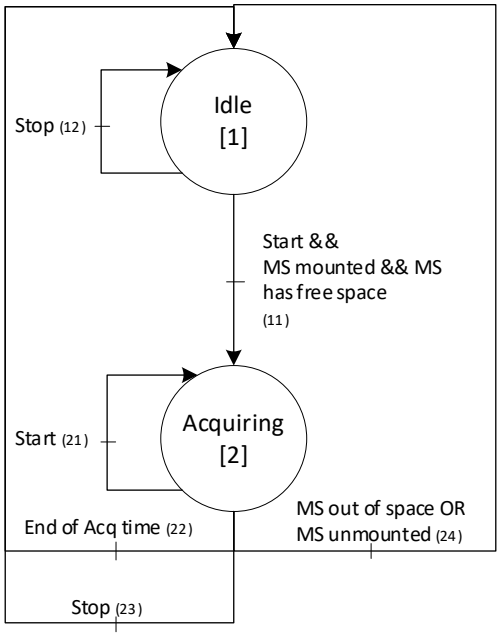

Fig: TSA Finite States Machine

Transition details & state information reporting:

11	Transition from Idle [1] to Acquiring [2] only on start signal, if MS is mounted with "enough" space. 'Enough' can be arbitrarily set to 100 MB. Once in state [2], report 'Acq State' as "Aquiring" and clear 'Acq State Additional Info'
12	Stay in Idle [1] on stop signal while in state [1] No update of 'Acq State' or'Acq State Additional Info'
21	Stay in Acquiring [2] on start signal while in state [2] No update of 'Acq State' or'Acq State Additional Info'
22	Transition from Acquiring [2] to Idle [1] on end of acquisition time timer. Once in state [1], report 'Acq State' as "Idle" and set 'Acq State Additional Info' to "run <run id> ran to completion"
23	Transition from Acquiring [2] to Idle [1] on user Stop signal. Once in state [1], report 'Acq State' as "Idle" and set 'Acq State Additional Info' to "run <run id> stopped by user"
24	Transition from Acquiring [2] to Idle [1] on device unmounted signal or device out of space. Once in state [1], report 'Acq State' as "Idle" and set 'Acq State Additional Info' to "run <run id> stopped due to MS device unmounted or out of space"