
CERN

LEBT Settings Protection

Proposal to use SIS with UCAP

Tibor Bukovics
06/05/2020

Challenges with the LEBT protection

1. The protection has to **validate settings** according to **different operation modes**. Each mode has different limits defined for each device.
   (OP, LOW_INT, MD)
2. The protection has to count unsuccessful validations for a predefined **time window** (currently 1 hour) and block operation if the counter exceeds a limit.
3. The protection has to follow **how long the system** is running **in each operation mode** and **lock** if it's running **longer than a predefined limit**.
4. Counters should be controlled though an LSA virtual device.
5. The protection has to act on the CHOPPER BIC, which already has OTHER SIS logic channels.

- Proposal to use UCAP converters together with SIS.

What is UCAP

- UCAP = Unified Controls Acquisition and Processing Framework
- ➢ UCAP very similar to SIS, it collects data from the system, organize, and executes user code to process this data.
  - Lightweight and modular.
  - It has wide range of event triggering.
  - **Good support for testing and debugging.**

- ➢ UCAP can only publish the processed data trough RDA3 virtual devices.

- ! UCAP has no support to interact with JAPC or LSA, not possible to set devices.

SIS with UCAP

Mixing the best from two worlds:

- With UCAP, the user code could be developed by **full IDE support** in pure java. Also the code could be verified by **automated testing** and can be used in demo mode.
- It's possible to create multiple independent converters, which are responsible for a simple task. This results in **lightweight code**, which is **simple to maintain, test and debug.**
- Each UCAP converter **can be triggered separately**.
- UCAP converters can be monitored and logged through RDA3 subscriptions.
- With SIS, a logical decision tree can be built between the UCAP converters.
- SIS **battle tested code** responsible to final decision making and action.
- UCAP could also serve as a DEMO data generator for integration testing with SIS.
- **SIS can be kept simple.**

SIS with UCAP

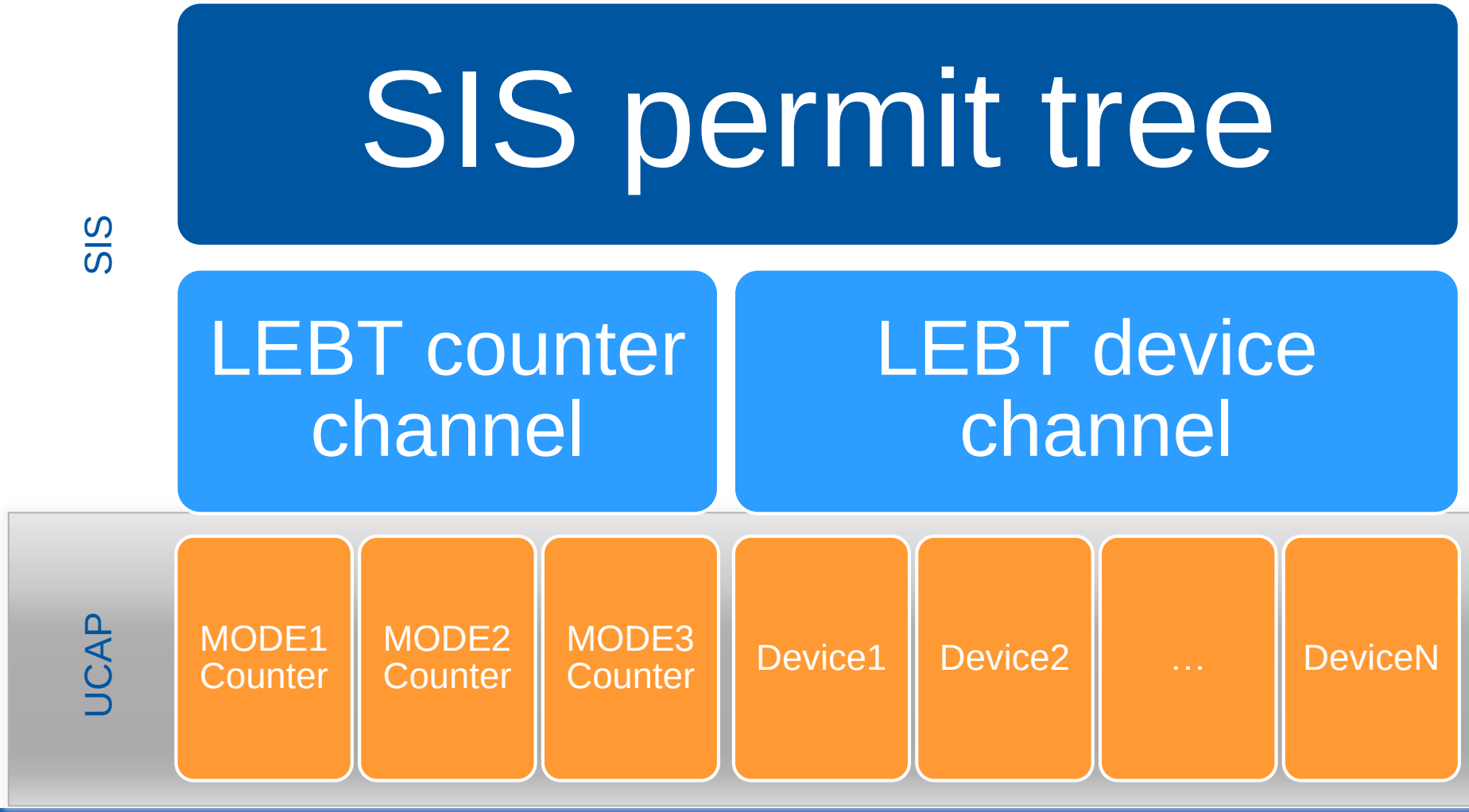
The idea:

- Use lightweight UCAP converters on top of each monitored device AND keep SIS pure and simple, focused on LOGIC.
  1. UCAP "**device**" converters are processing the data and validating according to the currently selected mode. If the device is in the right range UCAP virtual device publishes a TRUE permit.
  2. For each operation mode, there is a "**counter"** UCAP converter. They are monitoring all the device converters and if a device converter publishes a FALSE permit for the current mode, the counter permit stores this event in it's internal list. If the number of events on the list exceeds a predefined limit, the counter permit publishes a FALSE permit.

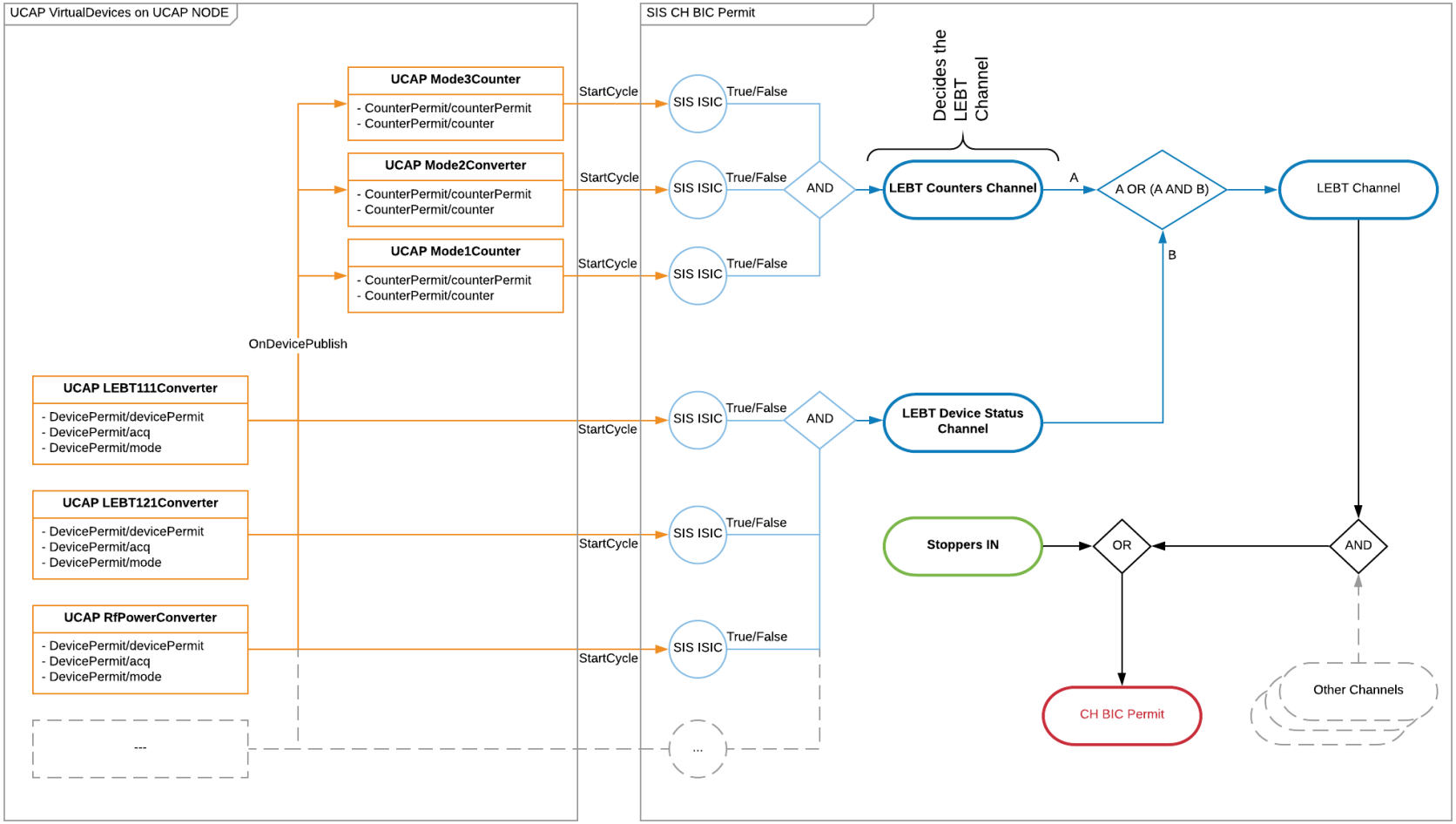
  - All UCAP converters process immediately when a source device publish new information. This guarantees the fastest status refreshment.

- All these converters are monitored from SIS.
  - The SIS logic evaluation is triggered by the **start cycle** signal.
  - SIS logic tree fulfills to block the BIC permit only if the counter UCAP converter publish false permit.

SIS with UCAP: Structure

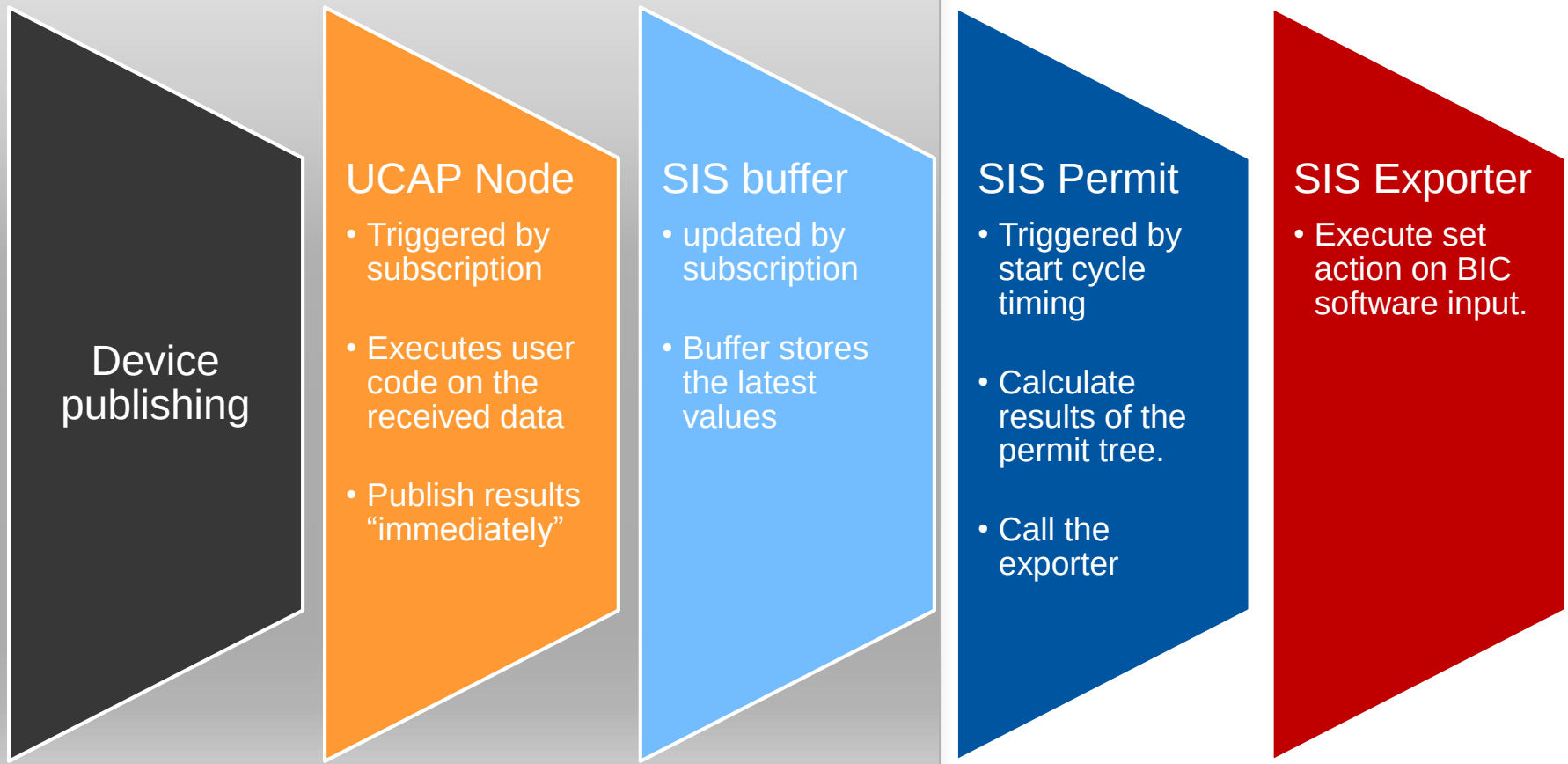


SIS with UCAP: Structure



SIS with UCAP: Timing

Before the start cycle, not fixed time.



Failover protection

- ✓ **Device not available or in exception:** UCAP converters are designed in a way to verify their subscriptions data and catch exceptions, and publish exception to RDA3 subscriptions.
- ✓ **UCAP node dies**: If the virtual device of a converter is not available SIS value condition fails with exception. Permit tree evaluated with false for that data.
- ✓ **UCAP converter does not publish**: For each UCAP converter subscription SIS checks the timestamp of the last message. If it's older than 1300ms SIS value condition on that subscription will fail.
- ✓ **UCAP counter reset on restart**: UCAP counters are stored in the node memory, which would be erased in case of node restart. This is a known risk but the converters can detect this situation and execute further action.

- ○ **UCAP node management access right:** each UCAP node configured with unique RBAC right. All critical management commands (converter configuration, stop, start, delete etc.) protected by this RBAC.

Pro and Contra

Pro	Contra
Utilizes the best from both worlds. SIS code can be very simple and focusing on the logic.	Everything can be implemented in SIS only. This design introduces one extra layer and increases the number of points of failure.
Node restart can be detected and signaled to SIS to block operation. Maybe file restoration will be possible in the future.	Counters reset when UCAP node restarts. However, this would be the same case for SIS.
Adaptive event triggering, UCAP processes as soon as data available. SIS permit executed only at start cycle.	This design would use more the CMW (more subscriptions, more publishing) and this is an extra step in the data transmission.
Distributed data processing.	Operators also need to understand that there is an another system which should be monitored.
User code can be simple and easy to debug. Also can be validated with automatic tools.	

Why should then use UCAP

- UCAP has strong support from CO and it's an evolving product.
- The unit testing and debugging capability helps to deliver more reliable quality user code.
- A java developer can learn UCAP in two weeks and easier to maintain the pure java code converter.
- Because UCAP converters acts as soon as a device publishes, technically there is no delay from this multiple layers. When SIS permit triggered by the start cycle, it has all the data already.
- Processing time reduced on SIS side.

Schedule for release

- A proof concept of the converters, their unit tests and SIS logic already exists and can run on DEMO data.

- Final version of the SIS code and the UCAP node could be released **within two weeks** upon request.

- To make a similar logic in SIS only would take upon request 3-4 weeks.