Proposed Export Service Refactor

For Delhi (or later)

Premise

- Export Distribution is essentially an EAI (enterprise application integration) problem
- Going forward, having one export service will not scale
  - It will contains too much code for all the possible formats, transformations, endpoint, etc.
  - Most of the code is not used in any single deployment
  - The copying of messages per client will not scale either
- Client registration may not be dynamic & and could be done by configuration
- End users need something simpler

What does export do anyway?

- Export Distribution is about taking incoming Events/Readings and:
  - Validating the data in the Event/Reading (valid device, valid readings, etc.)
  - Filtering (per device or value descriptor today, but that could be open to other filters like temps between a range, etc. later)
  - Transforming the data to the preferred format (XML, JSON today but could be YAML, TOML, CSV, etc.)
  - Perform other transformation operations (optionally compressing it, optionally encrypting it today)
  - Delivering the data to the endpoint of choice via protocol of choice (which is MQTT, MQTTS, HTTP, HTTPS, cloud, Rules Engine, analytics provider etc.).
    - This could be on box (to another service)
    - This could be off box (to cloud or enterprise or on prem server)

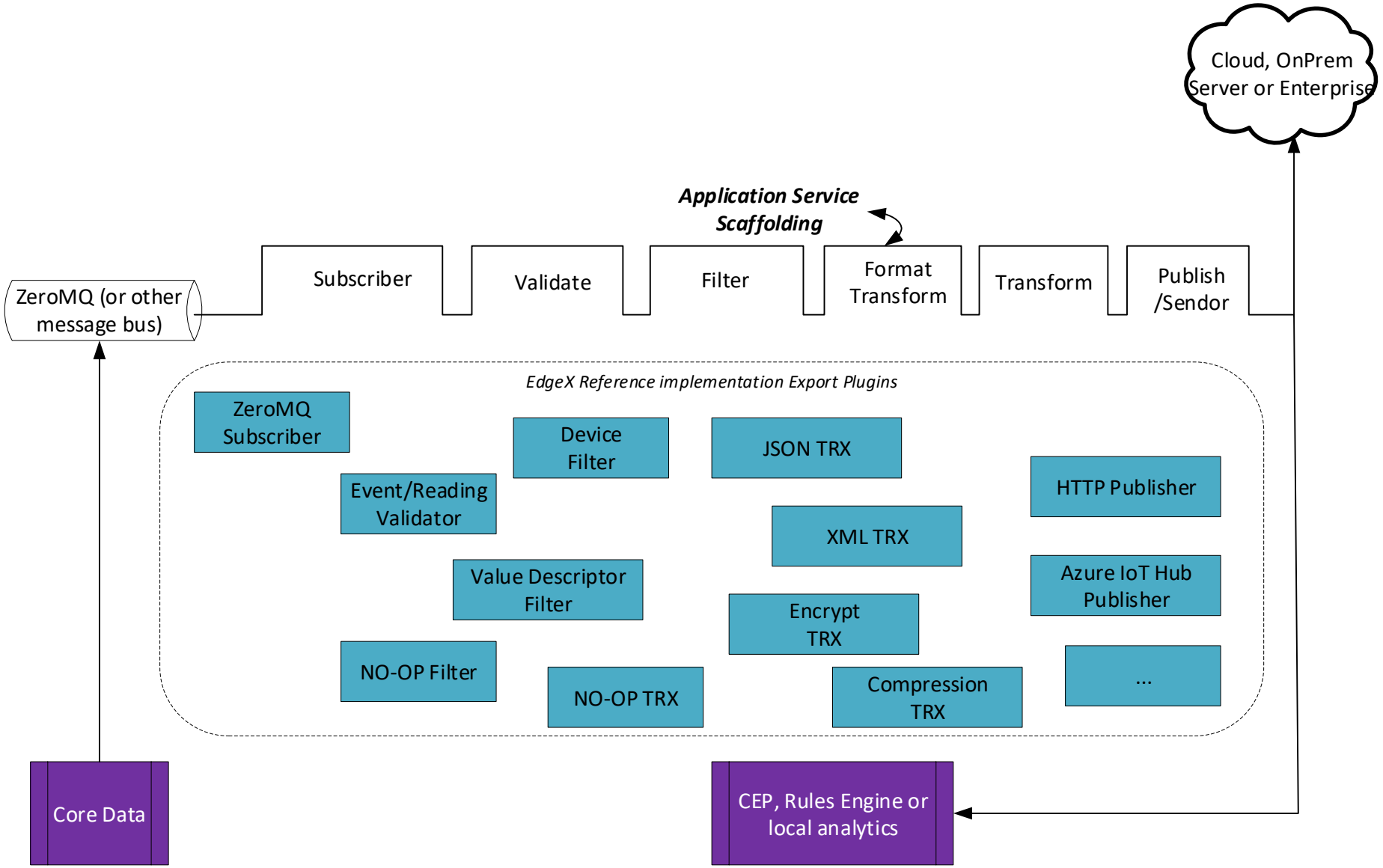
The Proposed New Export Service

- Develop a scaffold template or “SDK” that allows the easy creation of new export distribution services
  - The scaffolding keeps the message data moving through a pipe/filter system
  - Modules provide the validation, filtering, transforming, etc.
  - New export services are created from the scaffolding and a selection (or custom created) modules
- We would need to define the API, input and output of each module (much as EAI does) so that they are easily connected into the scaffolding.
  - Much of this is already done with our existing export
- We would define some reference implementations of many of the modules
  - For transformation, filtering, etc.
  - For endpoint distribution to Azure, AWS, etc. (not unlike how we provide Modbus, BACnet, etc. device services today).
  - We would have no-op implementation module that do nothing for that stage of work
- The “SDK” could be done in multiple languages in the future.
  - Any existing EAI tool could be used to provide the scaffolding and modules when it exists (example: Spring Integration when using Java).

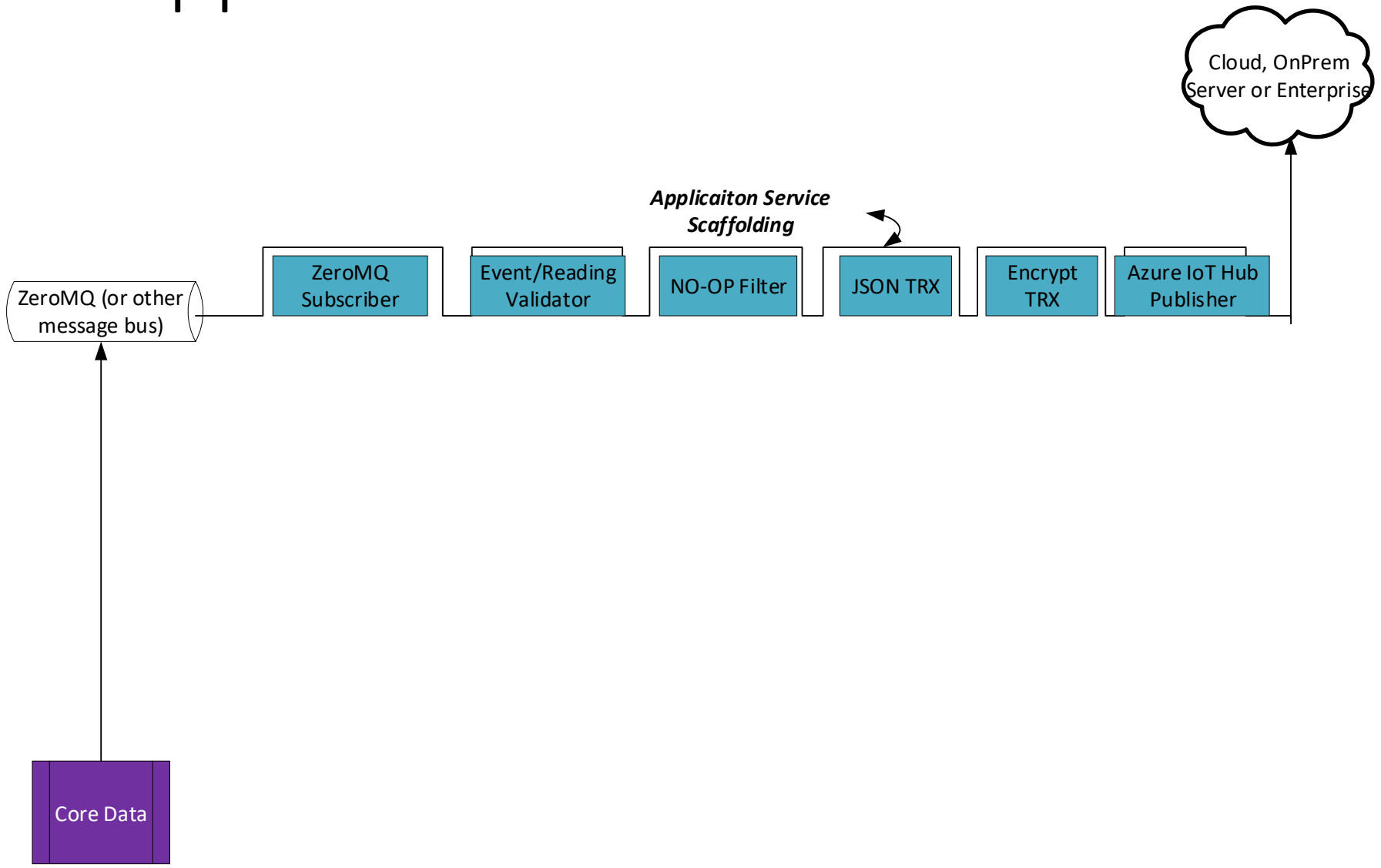
“Application Services”

- In the future, there is not one export service, but many export services – one per export use case
  - I recommend we call these “application services” that facilitate connectivity to various application needs
  - Not unlike how device services provide connectors to devices/sensors
- The export client service becomes optional
  - We configure the application service
  - Optionally the users or community could build export clients to facilitate the dynamic change to an application service where necessary
- The old Export Services could co-exist with the new export services until we are ready to retire it.

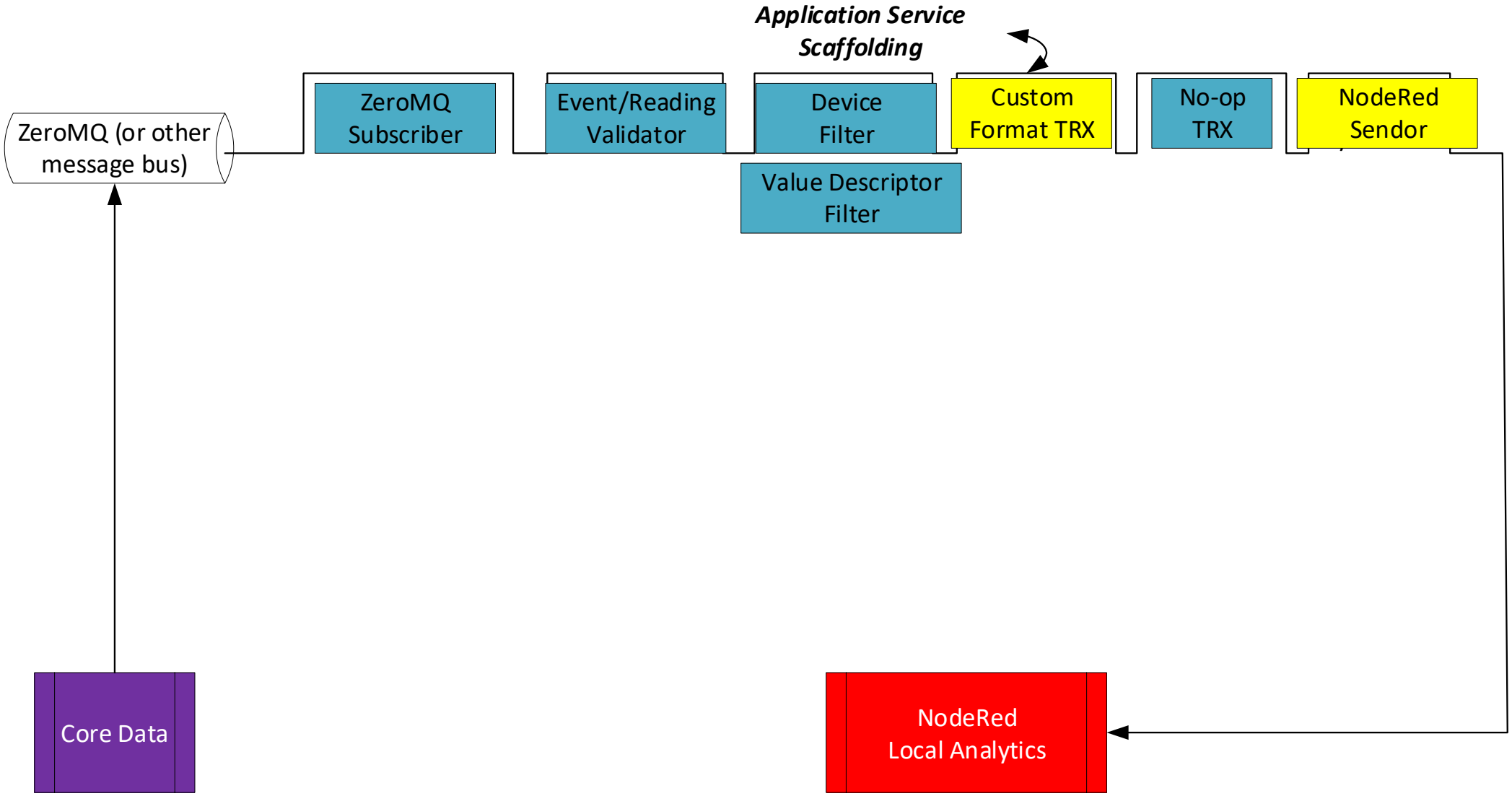
Proposed Application Service Design



Example A Application Service



Example B Application Service



EdgeX with multiple Application Services