**FORMATO DISEÑO Y DESARROLLO**

1. **INTRODUCCIÓN**
   1. **Propósito del sistema**

Implementar un flujo de procesamiento y disponibilización de datos, para el flujo de trabajo por lotes de múltiples fuentes.

* 1. **Alcance**

La solución propuesta está compuesta por varias historias de usuario, que comprenden el tratado de la información, el procesamiento y la depuración de los datos entregados de fuentes ARMDEF:

* **5521 HU - Configurar punto de servicio:** Historia por la cual se configura todo lo respectivo al punto de servicio.
* **5822 HU - Configurar dispositivo medidor eléctrico:** Historia por la cual se configura todo lo respectivo al medidor eléctrico.
* **10693 HU - Asociar/desasociar medidor eléctrico principal a punto de servicio:** Historia por la cual se realiza la asociación de un punto de servicio a un medidor eléctrico, esto por medio de interfaz.
* **13627 HU - Configuración de Sistemas externos:** Historia de usuario por la cual se realiza la configuración de la conexión a sistemas externos de forma que el sistema pueda hacer intercambios de información desde y hacia dichos sistemas.
* **5070 HU - Ajustes de Cortes de Energía**: Historia por la cual se realiza la configuración de llenado de datos en caso de apagón.
* **4731 HU - Completar datos faltantes**: Dependiendo de la configuración registrada en la aplicación, en caso de apagón y perdida de lecturas se debe rellenar aquellos vacíos generados, con nulos o ceros.
* **5191 HU - Configurar variables de perfil:** Historia por la cual se realiza la configuración de las variables de perfil de carga.
* **5193 HU - Configurar variables de registros:** Historia por la cual se realiza la configuración de las variables de registro.
* **5194 HU - Configurar variables de Eventos:** Historia por la cual se realiza la configuración de las variables de eventos.
* **4715 HU - Validar etiqueta de variable**: Dada la configuración de variables, se debe almacenar o no los valores de las lecturas que llegan en dichas variables, según lo marcado en la configuración.
* **5144 HU - Umbrales de validación de lecturas:** Configuración por la cual se define el rango de fechas en pasado y en futuro que se deben procesar.
* **4746 HU - Validar lecturas en pasado y futuro**: Dada la configuración de umbrales, se debe realizar una depuración de las lecturas cargadas que no cumplan el rango de fechas en pasado y futuro.
* **4752 HU - Completar información de lecturas**: Identificar la pertenencia de las lecturas y realizar todo lo requerido para poder completar las lecturas de forma de que puedan pasar por el procesamiento por lotes.
* **14177 HU - Versionar lecturas:** Proceso por el cual se almacena y versionan las lecturas.
* **6207 HU - Ejecutar validación de intervalos:** Historia que describe la forma en que se debe realizar la validación de los intervalos para lecturas de perfil de carga.
  1. **Información general**
* *Project name:*
* *Activity/Bug ID/Functionality:*
* *Source Repository:*
* *EA Path*
* *Toggle Feature Name*
* *Product Analyst:*

1. **DISEÑO DETALLADO**
   1. **Vista de Proceso**

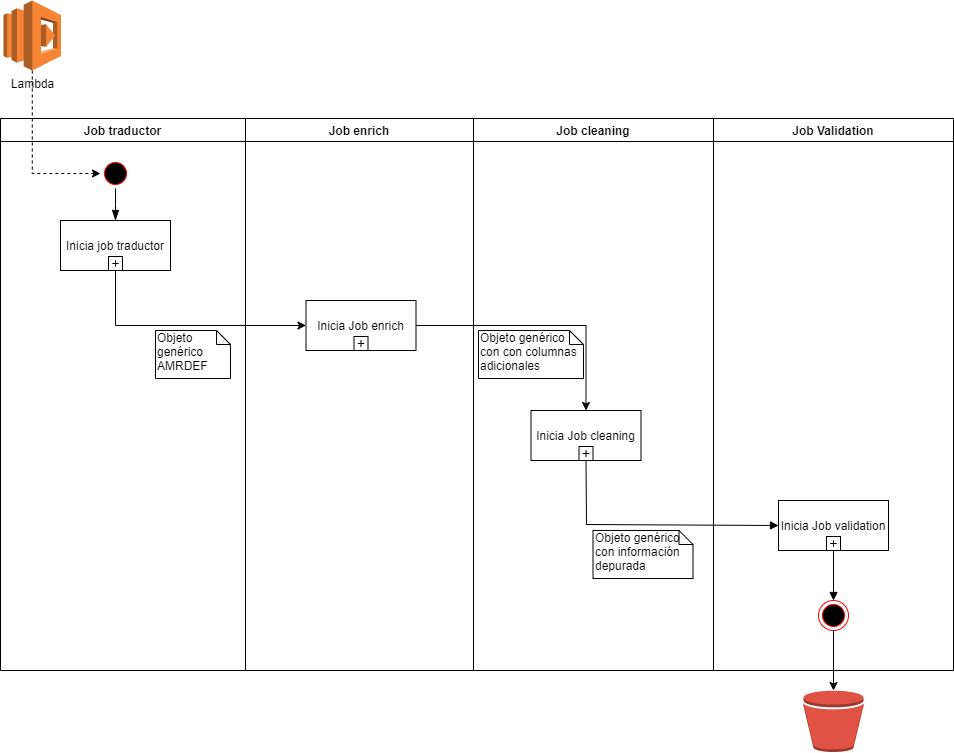


Diagrama 1- Vista general de proceso

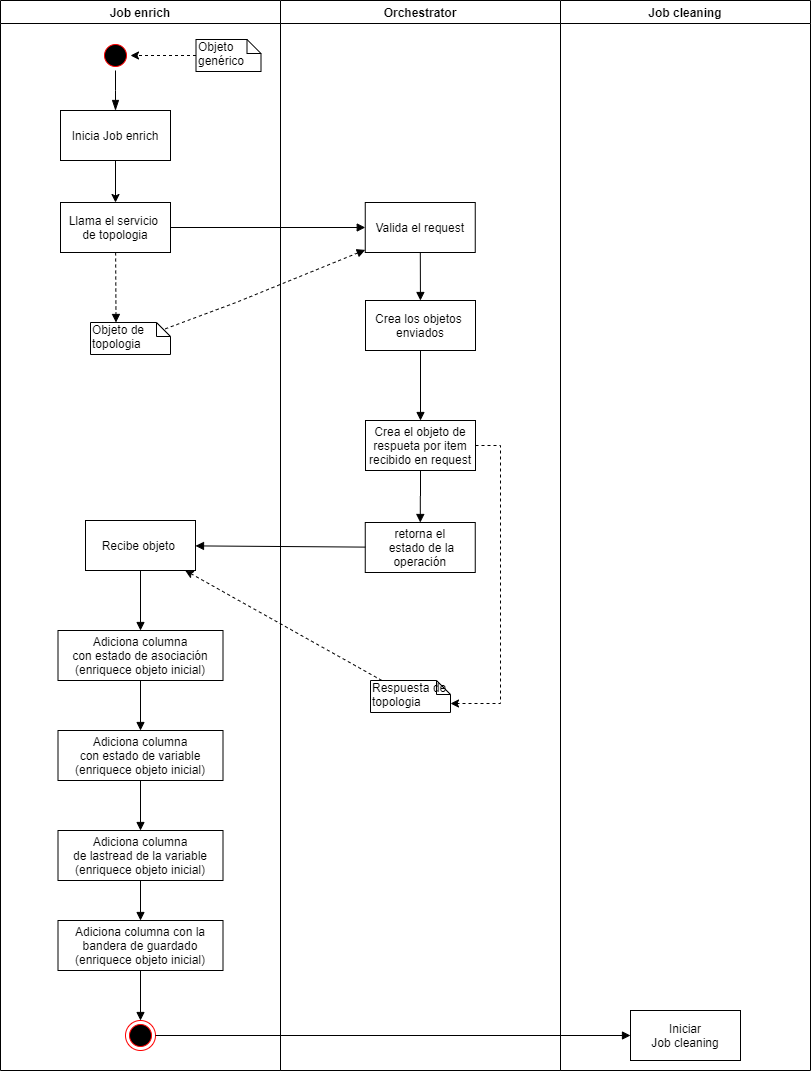
****

Diagrama 2 - Vista de Job Enrich

**Notas:**

* **Inicio Job Enrich:** Se aclara que para este proceso se usara nada más los objetos para la gestión de la topología del dispositivo.
* **Objeto Genérico:** Se debe tener en cuenta que el objeto genérico inicial, es decir el que inicia el proceso de enriquecimiento no cuenta con muchos de los campos mostrados en los ejemplos, sin embargo el proceso de enriquecimiento podrá trabajar con el mínimo posible.
* **Respuesta Topología:** Se debe tener en cuenta que la respuesta entregada por el servicio de topología será la misma que el objeto enviado originalmente, con la única diferencia que se agregan las propiedades bandera para indicar el estado de alguna transacción.
* **Orchestrator:** Se debe tener en cuenta que para hacer las peticiones a este servicio, se debe transformar la información de tabla a JSON, esto para poder hacer una petición valida. De igual forma la información de respuesta será entregada en dicho formato para ser agregada como columnas al objeto genérico inicial.
* **Estampa de tiempo UTC:** Las lecturas contaran con la estampa de tiempo en la que recolecto los datos la fuente. Sin embargo esta estampa es capturada en la hora local, por lo tanto la estampa de tiempo UTC debe ser calculada. Para esto se debe tener en cuenta la zona horaria en la que se encuentra ubicado el medidor, dicha configuración será retornada junto con el proceso de enriquecimiento. <https://earthsky.org/astronomy-essentials/universal-time>

**Medidor ubicado en Bogota – Colombia (UTC-05)**

**Hora de lectura local: 10:00**

**Hora UTC: 15:00 (03:00PM)**

**Calculo: 10:00 + 5**

**Medidor ubicado en Buenos aires – Argentina (UTC-03)**

**Hora de lectura local: 12:00**

**Hora UTC: 15:00 (03:00PM)**

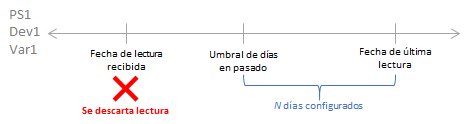
**Calculo: 12:00 + 3**



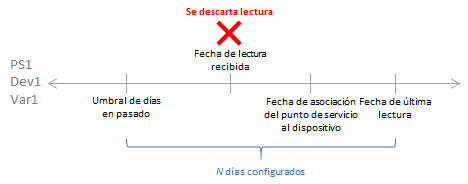
Diagrama 3 - Vista de JobClean

**Notas:**

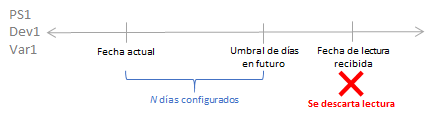
* **Descartar por pasado futuro:** Para descartar las lecturas se cuentan los días hacia atrás configurados. En caso de futuro se toman desde la fecha actual para adelante. La configuración de los días para realizar este descarte serán retornados desde el servicio de plataforma get\_configuration. Se debe tener en cuenta los siguientes caso para el descartado de lecturas por pasado futuro:
  + Se reciban lecturas con fecha menor a la fecha de última lectura menos los días configurados entonces el sistema deberá descartar dichas lecturas.



* + Se reciban lecturas con fecha menor a dicha fecha de asociación entonces el sistema no evaluará los umbrales de almacenamiento en pasado sino que las descartará automáticamente.



* + Se reciban lecturas con fecha mayor a la fecha actual más los días configurados entonces el sistema deberá descartar dichas lecturas.

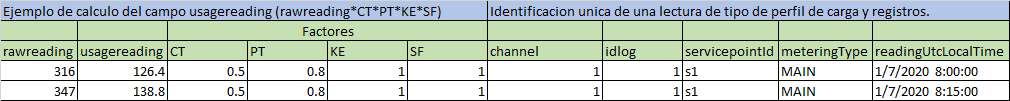


**Nota:** La información de última lectura y fecha de asociación son retornadas durante el proceso de enriquecimiento en la sección de topología.

* **Descartar por duplicados en memoria:** Si en las lecturas se encuentras dos lecturas para la misma variable, estampa de tiempo y log pero con diferente valor, se debe descartar por inconsistencia.
* **Consulta para recuperar lecturas activas en TrustData:** Se debe buscar la menor fecha de lectura en el bloque de memoria.
* **Rellenar lecturas de LP:** Se debe rellenar con ceros o nulos dependiendo de la configuración después de encontrar la bandera de apagón.
* **Finalización de proceso:** Al finalizar el proceso se deben remover las columnas banderas creadas por el enriquecimiento, es decir se debe finalizar con el objeto genérico original de topología con lecturas depuradas. Las banderas que deben ser removidas son:
  + servicePointVariable\_\_lastReadDate
  + StatusDevice
  + StatusVariable
  + SavedFlag
* **Versionar LP y Registros:** Cuando la lectura es nueva se carga con versión 1, si la lectura coincide con algún registro de **TrustData** pero con diferente valor de lectura entonces es marcado con el siguiente número consecutivo y es almacenada, es decir que se inserta un nuevo registro sin modificar el anterior sin embargo con un numero de versión diferente de forma que se tengan almacenadas todas las versiones de lectura. Se copia los valores de estimado, editado y validaciones de la versión anterior. Para realizar el versionamiento de las lecturas de tipo perfil de carga y registros se debe tener en cuenta alguna los siguientes criterios:
  + En caso de ser una lectura nueva se crea la versión "Original" que cuenta con las estampas de tiempo tal y como se obtienen de las fuentes de datos.
  + En caso de que existan las lecturas sin embargo las magnitudes sean diferentes, se crea la versión “Original n+1”, siendo “Original n”, la más reciente.
  + En caso que se encuentren lecturas nuevas y existentes para variables tipo perfil o registro, el sistema obtiene la versión "Usage" a partir de la versión "Original" más reciente.

Magnitud versión "Usage" = Magnitud versión "Original" \* Factores (CT, PT, KE, SF)

* Ejemplo de cálculo de usage e identificación de lecturas



Los campos que hacen unica la lectura son:

* + Channel
  + Log
  + ServicePoint
  + MeteringType
  + readingUtcLocalTime

Para el calculo del usagereading es igual a rawreading\*CT\*PT\*KE\*SF.

* **Completar datos faltantes (rellenos de nulos y ceros)**: Los siguientes son los escenarios para rellenar ceros y nulos en lecturas de perfil de carga:
  + **Para cortes de energía**: Se debe tener en cuenta que el servicio que obtiene la configuración de ceros y nulos, también retornara la configuración del rellenado correspondiente para varias banderas, que deben ser comparadas con el campo del objeto genérico llamado **validationFlags,** este campo contiene la información de los QualityCode en formato JSON. Se retorna toda la tabla de configuración de apagón (PowerOutage) para su comparación en cada uno de los registros, ya que los QualityCode pueden variar entre cada una de las lecturas, esta configuración será retornada por el servicio get\_configuration.
  + **Sin cortes de energía**: Se debe completar los intervalos con el valor de lecturas en nulo para los intervalos faltantes.



Diagrama 2 - Vista de Job validation

**Notas:**

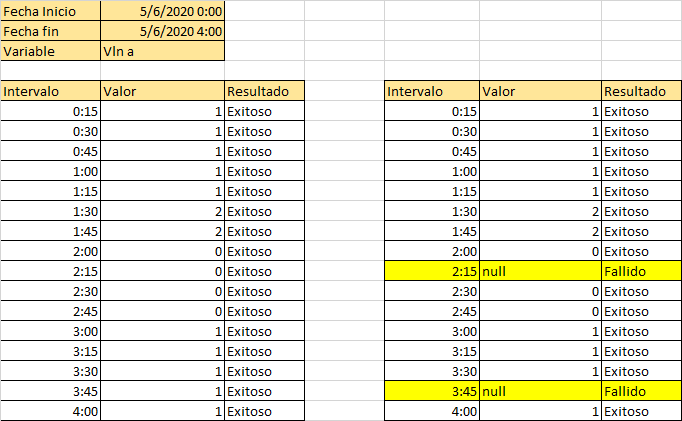
* **Valida datos cantidad de datos descargados contra la cantidad estimada:** Se debe tener en cuenta que en esta acción se agrupa únicamente por variables, es decir que la validación de estampas de tiempo faltantes se hace por cada variable que se encuentre en las lecturas. También se debe tener en cuenta que las operaciones siguientes se realizan por variable hasta que se envié el resultado de la operación. El cálculo de esta operación es de la siguiente forma:

𝑁° 𝑑𝑒 𝐷𝑎𝑡𝑜𝑠 esperados = 𝑁° 𝑑𝑒 lecturas 𝑝𝑜𝑟 ℎ𝑜𝑟𝑎 ∗ 𝑁° 𝑑𝑒 ℎ𝑜𝑟𝑎𝑠

𝑁° 𝑑𝑒 ℎ𝑜𝑟𝑎𝑠  = 31 horas

𝑁° 𝑑𝑒 𝐷𝑎𝑡𝑜𝑠 = 4 ∗ 31 = 124 datos

* En este proceso solo se usara el nodo de lecturas del objeto genérico.
* Es importante mencionar que un bloque de lecturas puede o no estar completo, en caso de no estarlo se debe identificar en donde se encuentra el vacío para marcarlo como lectura fallida.



En el ejemplo anterior la tabla de la izquierda representa un paquete de lecturas donde todas son correctas, es decir que se marcaran como exitosas. En la tabla de la derecha se ve que existen dos vacíos marcados de color amarillo, en este caso se crearan los intervalos faltantes con valores nulos y serán marcados con estado fallido, los demás se marcaran en estado exitoso.

* Las lecturas serán marcadas en el campo **usageValid**, con true o false dependiendo el resultado de la operación. En caso de ser fallido, adicionalmente se debe llenar el campo **ValidationDetail** especificando el fallo, con el siguiente formato JSON:

{

"Usage":

{

"IntervalsError":"Interval not exist"

}

}

* La información de lecturas será creada por defecto como no agrupadas, es decir que serán guardadas en el **TrustData** con el campo **IsGrouped** como false.
  1. **Vista de Lógica**

Se adjunta el diagrama general con navegación para entender el flujo base y los objetos usados dentro de cada una de las operaciones:



* 1. **Vista de despliegue**

N.A.

* 1. **Vista de Componentes**

N.A.

* 1. **Vista de Información**

Objeto genérico de topología en formato JSON, tendrá la siguiente estructura:



En caso de no tener los datos para llenar todas las propiedades del objeto de topología, se podrá especificar únicamente los campos obligatorios para el funcionamiento del proceso, dichos campos se encuentran en el siguiente ejemplo:



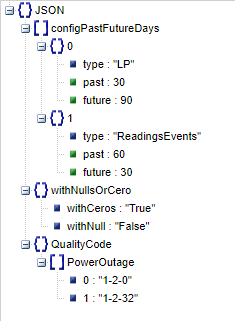
En caso de ser tabular cumplirá con la siguiente estructura, en el siguiente archivo encontrara la estructura tabular con formato enriquecido nombrado en el enrich:



La lista retornada desde los servicios de topología contara con una estructura similar a esta:



El objeto retornado por plataforma en la operación de Clean, es un objeto de configuración que contendrá todo lo respectivo ha pasado, futuro, nulos y ceros. La estructura de dicha respuesta será de la siguiente forma.



Se adjuntan las historias de usuario que tienen injerencia directa en el procesamiento, en este documento se encuentra el nombre de la historia, los criterios de aceptación y el log relacionado a la historia.



La definición de los logs junto con los campos que serán diligenciados para estos y los pasos en donde deben ser llenados, se especifica en el siguiente documento:



1. **Manejo de logs**

Generalidades:

* Mensajes relacionados a un flujo de negocio clave que desarrolla el sistema. Dentro del sistema PAP, se consideran logs de negocio aquellos generados durante el proceso de ingesta y procesamiento de información, en la capa transfer y processing and storage de la arquitectura. También se incluyen los eventos de negocio generados en la capa de servicios de negocio que atiende solicitudes de las demás capas de la arquitectura.
* Los logs deben ser codificados en UTF-8. Esto sigue las recomendaciones de *Common Event Format CEF.*
* Por convención, todos los logs deben estar en inglés – Estados Unidos
* Para un mismo tipo de log, por ejemplo, ReadginsProcessing, para un mismo HES, año, mes, semana, todos los procesamientos de lecturas serán trazados en el mismo archivo de log. Salvedad; PAP cuenta con un mecanismo de rollout de los archivos que permite evitar tener archivos de log de tamaños muy grandes.

**Estructura de logs Estándar (para todo evento)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Descripción** | **Tipo de Dato** | **Atributo JSON Sugerido** |
| Identificador de la Transacción | Permite la correlación de eventos. Es de gran ayuda para identificar a los usuarios o acciones que se han realizado sobre una operación, de forma que se puede tener una mayor claridad de la trazabilidad de los flujos y procesos del sistema. El identificador único debe ser definido por el API Gateway o por el primer microservicio que atiende una acción de un usuario. **NOTA:** Para logs de negocio en la ingesta y procesamiento, se propone que el transaction\_id sea el guid del archivo ingestado. | GUID | *transaction\_id* |
| Identificador del Usuario | Identificador único para el usuario. Este identificador debe corresponder a una propiedad que debe ser única por el usuario. Ejemplos: *nombre de usuario, correo electrónico, id de bases de datos.*  Cuando no existe un usuario relacionado al evento, se deja *System* por defecto. | Cadena. | *user\_id* |
| Identificador del Tenant | Es el identificador del dueño de la información, entendiendo como dueño la compañía dueña de los datos. Cuando el evento no está relacionado a un Tenant específico, se deja System por defecto. | Cadena | *owner\_id* |
| Estampa de Tiempo | Indispensable ya que permite el análisis sobre líneas de tiempo. Almacenar el tiempo en que sucedieron los eventos también permite conocer cuánto ha tardado una tarea específica en ser completada. | Fecha que incluye día, mes, año, hora, minuto, segundo y milisegundo.  Formato:  yyyy-MM-dd HH:mm:ss.ms | *timestamp* |
| Mensaje | Campo donde el desarrollador especifica un mensaje relacionando la operación realizada, su resultado o posibles eventualidades en el proceso. Es de gran ayuda para la realización de *debugging.* Se debe incluir todo el contexto del evento, teniendo la precaución de que **NO** es el mensaje de error comunicado al usuario. | Cadena. | *message* |
| Nivel de Log | Nivel de log el cual está sujeto a una escalada ordenada de mayor a menor importancia. | Numeración con los posibles valores mencionados en la sección *Niveles de Logs.* | *log\_level* |
| Tipo de Log | Tipo de evento o registro según el contexto operacional del software. | Numeración con los posibles valores mencionados en la sección *Tipos de Logs.* | *log\_type* |
| Fuente | Lugar en código donde se produce el evento. Debe incluir una ubicación específica con Nombre de la clase, método y módulo. | Estructura JSON la cual se sugiere que tenga alguno de los siguientes atributos.  *class\_id, method\_id, module\_id.* | *source* |
| (Opcional)  Categoría | Si type = security entonces otorga información adicional para propósitos de evaluaciones en seguridad. | Ver la sección Logs de Seguridad. | *De acuerdo con sección de logs de seguridad* |
| Si type = business entonces otorga información adicional para propósitos de negocio. | Ver sección de logs de negocio | De acuerdo con sección de logs de negocio |

Los logs de negocio almacenan información de eventos propios de la operación del sistema y que tienen valor para el contexto del negocio. A partir de lo anterior, y con el objetivo de continuar con una estructura estándar de los logs, se adopta el mismo enfoque de los logs de seguridad. Por ello, adicional al campo *Tipo Negocio,* debe agregarse un campo *“****business\_log****”* el cual debe contener una estructura JSON compuesta por una *Categoría y Contexto.*

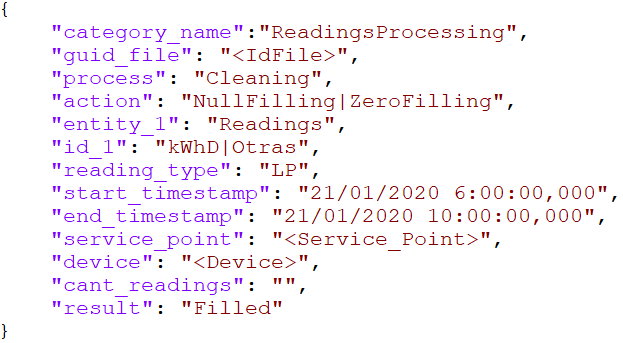
Para el caso particular del procesamiento de lecturas aplican los siguientes logs:



En el documento LogsPAP.xlsx, pestaña “Definición de Logs” se determina una estructura estándar para cualquier LOG de PAP y la definición de un campo Category el cual puede variar para los diferentes tipos de logs que aplican ( TransferToRAW, ReadingsProcessing, etc).

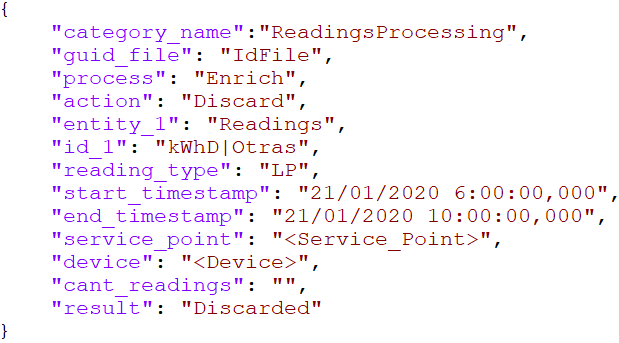
* **4731 HU - Completar datos faltantes (rellenos de nulos y ceros )**

Dada la necesidad de llevar la trazabilidad de las lecturas cuando el sistema complete lecturas entonces se deberá registrar en un log el Id de la variable, fecha y hora de la lectura, valor completado y motivo. Un ejemplo sería:



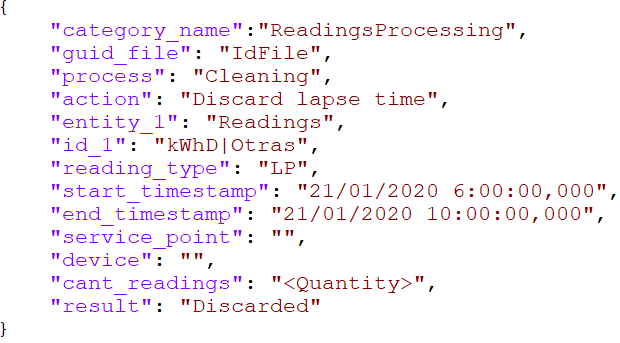
* **4715 HU - Validar etiqueta de variable:**

Dada la necesidad de llevar la trazabilidad de las lecturas cuando el sistema descarte lecturas entonces se deberá registrar en un log el Id de la variable que se descartó.



* **4746 HU - Validar lecturas en pasado y futuro:**

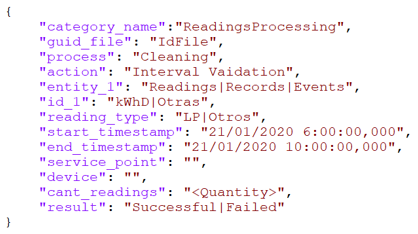
Dada la necesidad de llevar la trazabilidad de las lecturas cuando el sistema descarte lecturas entonces se deberá registrar en un log el Id de la variable que se descartó, la cantidad de lecturas descartadas y el motivo.



* **6207 HU – Ejecutar validación de intervalos:**

Dada la necesidad de validar intervalos, Cuando se ejecute el proceso de validación, Entonces el sistema deberá retornar el siguiente resultado:

* Exitoso: Si la cantidad de datos esperada es igual a la recibida.
* Fallido: Si la cantidad de datos esperada es menor a la recibida.

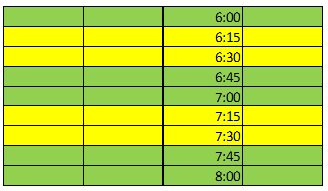


El campo Quantity denotará la cantidad de validaciones exitosas, si el resultado es Successful. El campo Quantity denotará la cantidad de validaciones fallidas, si el resultado es Failed.

Por otro lado, bajo el criterio:

* Dada la necesidad de explotar el resultado de la validación de intervalos
* Cuando se realice la validación, Entonces el sistema debe persistir la siguiente información para que pueda ser posteriormente consultada por otros procesos:
* Fecha y hora en que se realizó la validación
* Estampa de fecha y hora inicial del paquete de datos evaluados
* Estampa de fecha y hora final del paquete de datos evaluados
* Definición de la lectura evaluada (Punto de servicio, tipo de medida, dispositivo, fuente, versión, variable, etc.) (Ver HU 4752 vinculada)
* Cantidad de lecturas recibidas
* Cantidad de lecturas esperadas
* Diferencia entre datos esperados y datos recibidos
* Estampas de hora de los intervalos faltantes
* Resultado de la validación (Exitoso/Fallido)

Para cada tantos intervalos de validaciones fallidas como exitosas, se creará un evento de log, esto es:



Dado el escenario de la ilustración anterior en donde están resaltadas con verde las validaciones que fuerno exitosas y en amarillo las que fueron fallidas, es requerido registrar los siguientes eventos de logs para cada rango de intervalos así:

1. Intervalo de 06:00 Exitoso
2. Intervalo 06:15 – 06:30 Fallido
3. Intervalo 06:45 – 07:00 Exitoso
4. Intervalo 07:15 - 07:30 Fallido
5. Intervalo 07:45 – 08:00 Exitoso

Para un total de 5 eventos de log bajo este formato:



**Logs Tiempo de Procesamiento Jobs**

Para cada uno de los Jobs creados, se deberá crear un log de inicio y otro de finalización de log con el fin de mantener bajo control el tiempo de procesamiento de cada uno de ellos.

Nuevamente, nos basaremos en el mismo Log estándar describo en el documento LogsPAP.xlsx y para el campo Category utilizaremos el formato para *ReadingsProcessingLifecycle* así:



Este log permite conocer el momento en que inicia la ejecución de un Job o el momento en que termina de tal modo que un proceso de explotación de logs pueda determinar el tiempo total de ejecución de cada uno de los Jobs para cada uno de los idFiles.

1. **Decisiones de diseño**

* Se realiza una implementación en donde se separa cada proceso en Traductor, Enrich y Cleaning.
* En cada uno de los procesos se realizan las tareas pertinentes para cumplir su objetivo.
* La comunicación entre procesos se hará por medio de objetos genéricos organizados en columnas.
* La comunicación de las operaciones internas de cada proceso puede variar dependiendo de la necesidad, es decir que puede ser necesario consumir un endpoint que implicaría formatear la información en objetos JSON.