SOUR (SCM OPC-UA REDIS)

Progetto per l'implementazione del server OPC-UA di gruppo basato su REDIS per acquisizione dati dai PLC / MSIM

- ProductionLog - Bordatrici

Versioni

Le versioni rilasciate della documentazione e del server SOUR sono le seguenti:

Vers	Data	Descrizione
0.5	2018.07.11	Prima versione funzionante in debug e testata con client prosys
0.7	2018.07.11	Versioni successive con pulizia del codice (eliminazione componenti non necessari provenienti dal progetto MSim di partenza)
0.9	2018.07.12	Prima versione binaria per test come eseguibile console + log + traduzioni logparser...
1.0	2018.11.14	Completata documentazione
1.1	2018.11.17	Prima versione rilasciata per produzione con log standardizzato + documentazione x popolare area REDIS (1.1.1811.165)
1.1.1	2018.11.19	Integrata documentazione x Draft funzionalità
1.2	2018.11.26	Completata documentazione area REDIS e aggiornamento server (1.2.1811.180)
1.2.6	2018.12.06	Completata documentazione area REDIS e aggiornamento server (1.2.1812.185)
1.3.1	2019.01.31	Implementata gestione nuovi parametri in DataModel per SampleGroup, DeadBand, VisibilityGroup --> testing con FANUC/SIEMENS ed adapter CMS (1.3.1901.195)
1.3.2	2019.02.03	Review documentazione x definizione modelli tracciati JSON per log eventi/allarmi/produzione su base riunione 2018.01.30 (1.3.1902.196)
1.4	2019.02.18	Cambio Namespace, fix gestione allarmi
2.0	2019.03.08	Prima release 2.0 stabile
2.1	2019.04.12	modifica comportamento in avvio per sincronizzazione allarmi, sincronizzazione variabili e proprietà
2.2	2019.08.01	Prima release 2.2 con MQTT, fix comportamento invio allarmi 8gestione memoria persistente precedente invio stato allarmi), nuovo installer MSI

Terminologia

Iniziamo con la terminologia degli elementi che useremo nel seguito del documento:

Sigla	Definizione
REDIS server	E' un DB server di tipo NoSQL, hash based, di tipo InMemory Storage ad alte performance. Impiegato in questo progetto per il basso impatto sulle prestazioni e la flessibilità permessa di salvataggio informazioni da sorgenti diverse che vengono normalizzate in questo DB di normalizzazione.
Stecialized **ADAPTER**	E' il sw specifico per ogni UTE / tipologia di macchinario / controllo in grado di comunicare con il PLC/CNC/HMI e pubblicare i dati verso lo strato REDIS di disaccoppiamento
SOUR	E' il server OPC-UA del gruppo SCM che rende disponibili i dati pubblicati sul server REDIS tramite modalità OPC-UA a vari client in grado di consumare i dati stessi a valle del sistema.
GATEWAY	E' il componente HW impiegato per l'acquisizione dei dati resi disponibili dall'OPC-UA server SOUR.
Maestro Connect	E' la piattaforma cloud di SCM su cu sono raccolti e poi gestiti i dati prodotti dai dispositivi e macchinari IOT di SCM.

Principi di base

Il principio di base prevede che uno strato di cache intermedio sia inserito tra il generico adapter specifico per l'acquisizione dati da uno specifico PLC/CNC delle varie UTE di SCM e il server OPC-UA generico (ed unico) di gruppo.

Per poter disaccoppiare i due elementi, vista l'eterogeneità dei vari adapter ed HMI di gruppo, si è scelto uno strato di server in RAM di tipo NoSQL, in particolare REDIS.

Questo server è pensato per poter gestire senza problemi un notevole numero di scritture e letture che possono avvenire in modo contemporaneo.

Il server ha un bassissimo impatto su RAM e CPU del sistema e non garantisce persistenza su disco, quindi tutta la'rchitettura di scambio dati è progettata per NON considerare il dato trasferito come sempre disponibile.

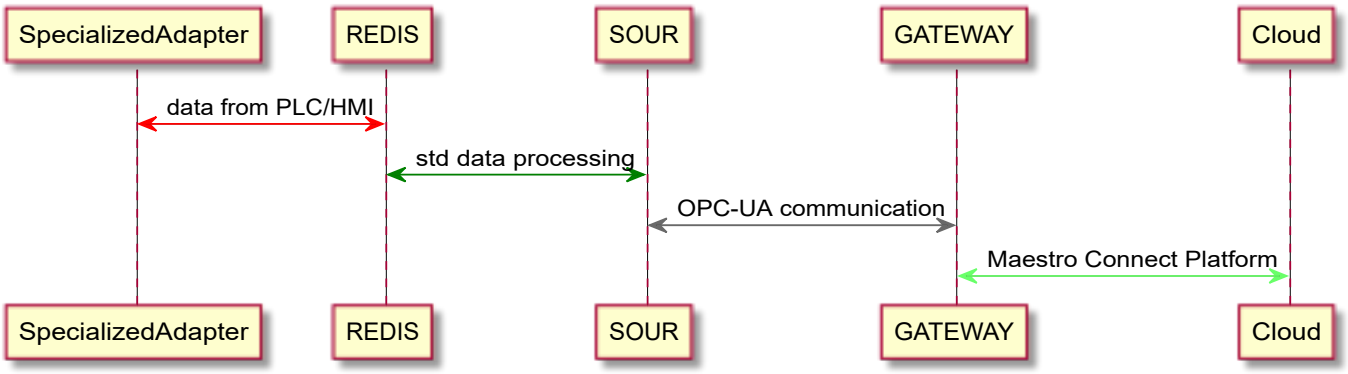
Sul server REDIS vengono salvate delle informazioni come "generiche variabili" che non sono specifiche di un singolo PLC/CNC ma che permettono di virtualizzare/normalizzare l'interfaccia seguente di comunicazione.

Lo schema di massima prevede quindi il seguente processo operati

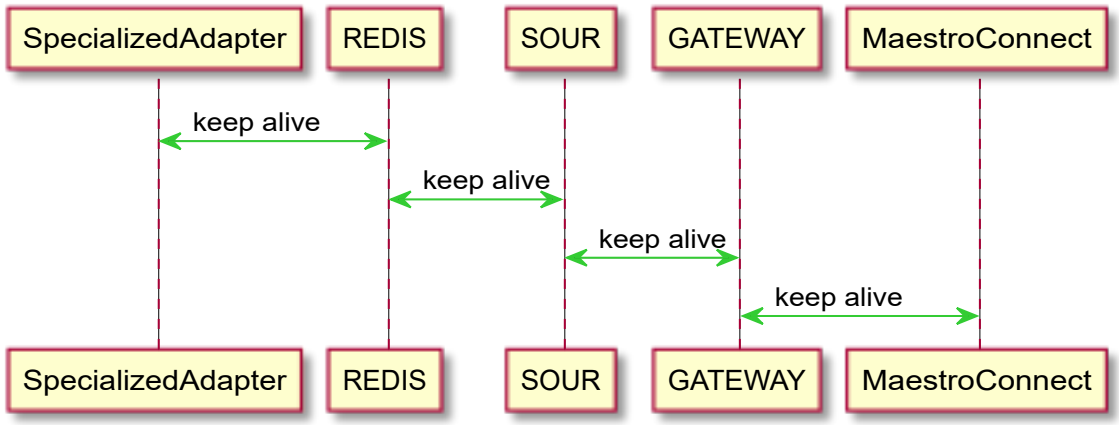
- ADAPTER pubblica informazioni DI STATO secondo logiche di timing/eventi concordate (watchdog ogni secondo)
- ADAPTER pubblica informazioni DI PROCESSO sul server REDIS secondo logiche di timing/eventi liberamente definite
- ADAPTER (o HMI) verifica informazioni pubblicate da altri (SOUR, GATEWAY, ...) sul server REDIS secondo logiche di timing/eventi liberamente definite
- SOUR pubblica informazioni DI STATO secondo logiche di timing/eventi concordate (watchdog ogni secondo)
- SOUR pubblica informazioni DI PROCESSO sul server REDIS secondo logiche di timing/eventi liberamente definite
- SOUR verifica informazioni pubblicate da altri (ADAPTER, GATEWAY, ...) sul server REDIS secondo logiche di timing/eventi liberamente definite

Schema di riferimento

Questo lo schema di riferimento generale del progetto



Aggiungiamo anche questa rappresentazione per i servizi di verifica e controllo reciproco in modalità watchdog / keep-alive.



Requisiti per installazione

Prerequisito epr il funzionamento è l'installazione di Redis per windows (versione compilata da Microsoft).

https://github.com/MicrosoftArchive/redis

ed in particolare la release del 2016

https://github.com/MicrosoftArchive/redis/releases

E' poi possibile aggiungere a questo anche l'installazione del client di gestione RedisDesktop:

- Redis Explorer: https://github.com/leegould/RedisExplorer
- Redis desktop (non free): https://redisdesktop.com/

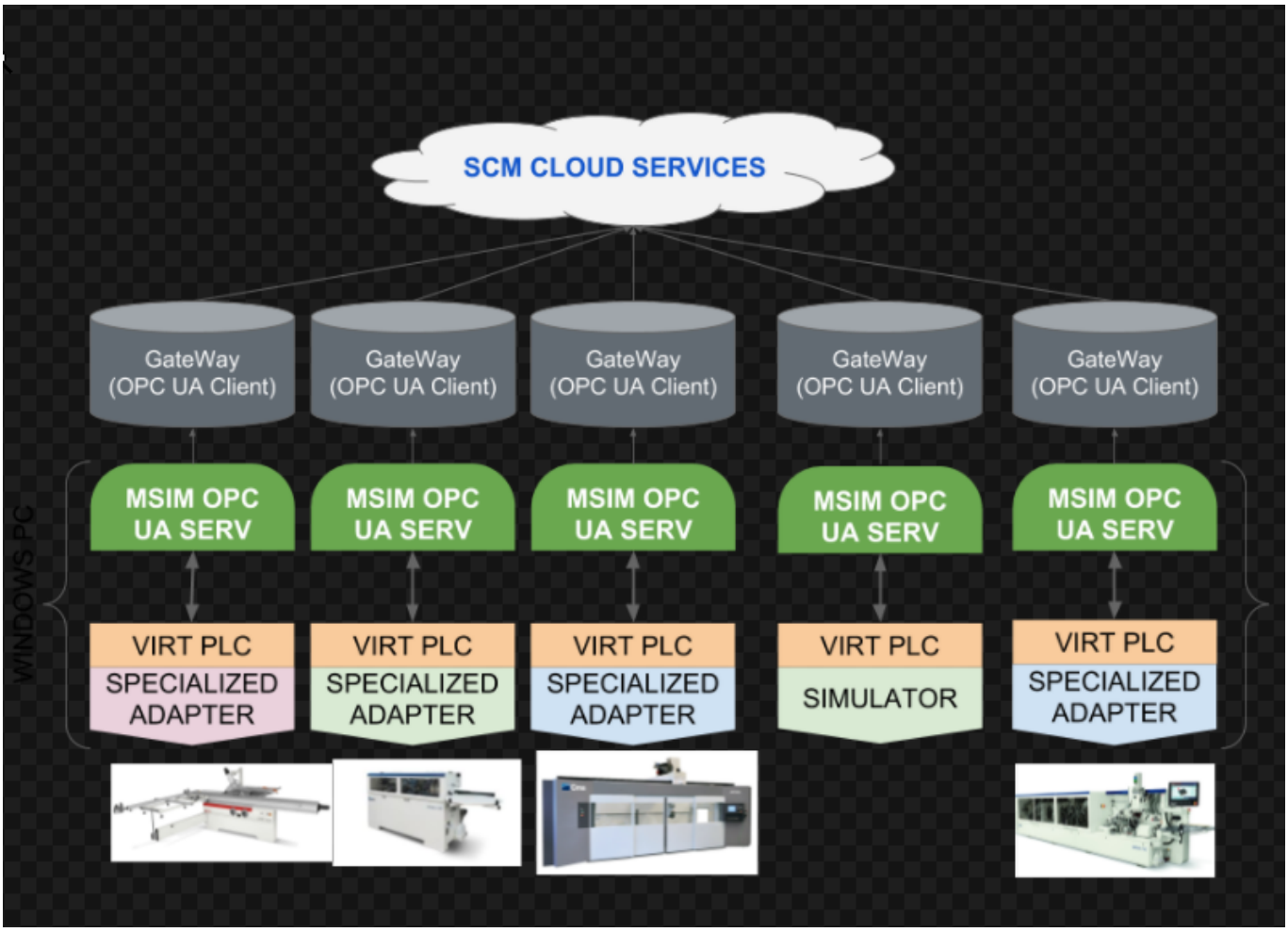
Regole di Naming

Le regole di naming, disponibili del doc di riferimento per il DataModel, si applicano in primis al DataModel dell'OPC UA server, e quindi a ritroso si applicano a tutte el variabili (che si popoleranno nel server REDIS e che poi il server OPC-UA pubblicherà)

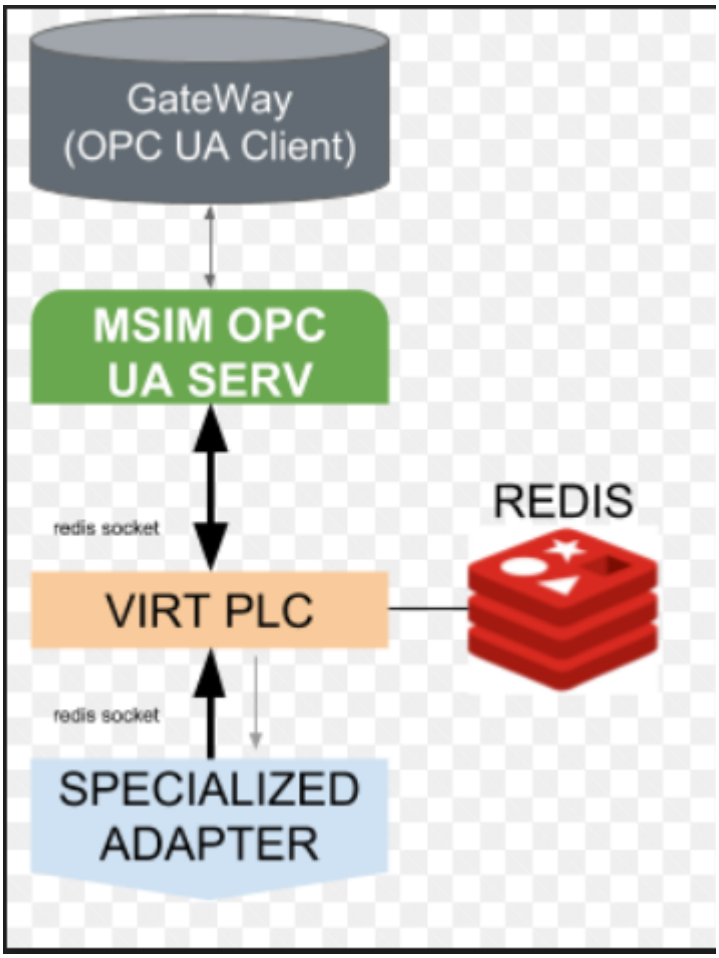
- Il carattere speciale **:** serve a definire la gerarchia implicita di raggruppamento degli oggetti in REDIS; e quindi va riservata a tale scopo (NON PUO' venire impiegato ad esempio come separatore in valori stringa da esplodere)
- PascalCasing: il nome delle variabili è di tipo PascalCasing con maiuscola iniziale
- Istanze numeriche con indici che partono da 01
- Gli ENUM devono essere scritti tutti in maiuscolo separati da underscore
- Nel caso di attributo opzionale non necessario per qualla particolare macchina eliminiamo il suo sottoalbero XML dal DataModel
- Nel caso di attributo mandatory e nel caso di non disponibilità del dato presentiamo il valore null. Utilizzato anche in fase di inizializzazione
- Nel caso di tipo Boolean devo scrive in REDIS come **true,false**
- Formato double con separatore punto, es **1.45467**
- Le date vanno inserite in formato ISO8601 UTC (**YYYY-MM-DDThh:mm:ss.nnnTZD**, es **2018-11-05T13:15:30.123Z**)
- Severità allarmi:
    - 900 Alarm
    - 500 Warning
    - 200 Info
    - 100 Maintenance
- Il separatore in caso di array è la virgola

Schemi di dettaglio

Una rappresentazione più specifica è fornita dagli schemi seguenti:

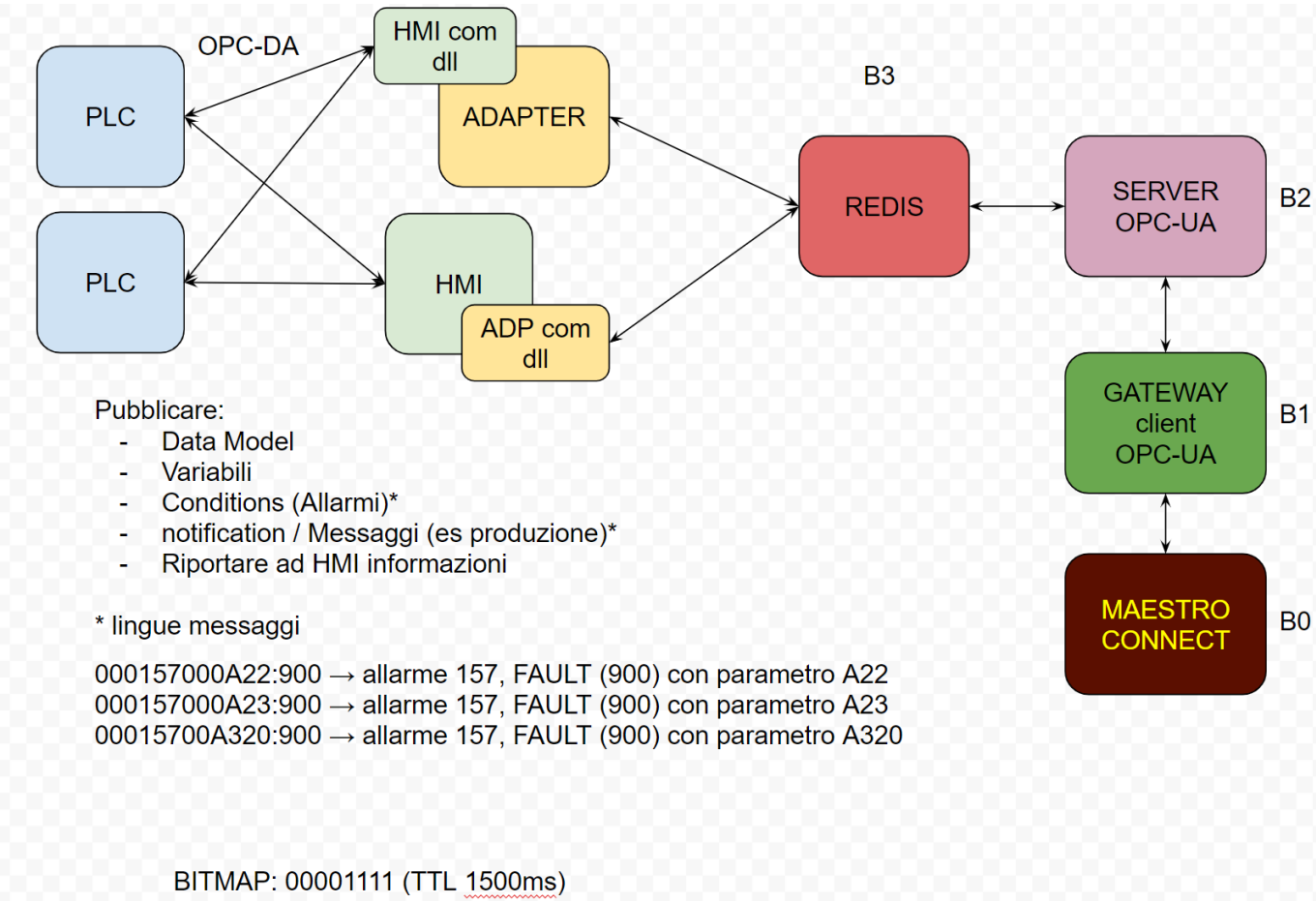


Più in dettaglio il sistema prevede che ci sia uno strato di normalizzaizone, che abbiamo qui definito VIRT-PLC, costituito da un server REDIS di disaccoppiamento.



In particolare quindi ogni tecnologia è responsabile della realizzazione della parte che arriva ad interagire (popolare i dati in ingresso dati e leggere dati dei sistemi che sono a valle di redis).

Più in dettaglio si possono avere scenari come i seguenti che verranno di seguito meglio descritti:



Dallo schema si evince che ci possono essere diversi scenari di implementazione che uniscono i PLC, le HMI ed il server REDIS, a seconda dei casi con impiego di dll di comunicazione incluse nelle HMI o al contrario producendo applicativi indipendenti dall'HMI che sfruttano modalità di comunicazione coi PLC basati su librerie tipicamente costruite per l'HMI stessa.

A valle di questo si trova il server REDIS che persiste le informaizone e fa da livello di strazione e normalizzaizone dei dati tra le diverse tecnologie (UTE) rappresentate sulla sinistra ed il server OPC-UA di gruppo (SOUR) sulla destra. Questo server poi interagisce con il gateway e questo con il cloud Maestro Connect del gruppo SCM.

Riferimenti importanti

Alcune note importatnti riguardo l'installer MSI di SOUR e il broker MQTT embedded in SOUR

MQTT

Dalla versione 2.2 è attivo un broker di pubblicazione MQTT verso il cloud.

Condizioni necessarie al funzionamento:

- Non sia presente la chiave di vet in REDIS all'indirizzo `SOUR:GwHw:Vers`, se vuoto/nullo = nessun veto, se presente il numero di versioni (o anceh solo una stringa non vuota) riferita al Gateway Hw viene inibito il funzionamento della sezione MQTT
- macchina attivata con SDK (quindi disponibili le informaizoni di user e pwd per il broker)

MSI installer

Aggiunto il progetto di generazione installer con wix, il processo jenkis crea l'installer come ultimo step di deploy

Modalità di installaizone file installer msi:

```
SOUR.Setup.msi INSTALLFOLDER=C:\IOT\SOUR /quiet
```

Elementi della soluzione

Di seguito sono indicati i vari elementi in gioco nel sistema e le specifiche di impiego e funzionamento.

SPECIALIZED ADAPTER --> REDIS SERVER

Questo primo componente andrà a popolare le informazioni secondo le PROPRIE esigenze, dettate dalle specifiche tecniche del PLC/CNC/HMI; e quindi rispettando i limiti tecnologici e i limiti di non invasività rispetto agli altri applicativi in esecuzione. ne.

OPC-UA SERVER --> REDIS SERVER

Questo è il componente comune per tutto il gruppo che è agnostsico dell'effettiva tecnologia. Con un timing indipendente dallo strato di acquisizione dati dell'adapter, il server procede a leggere le informazioni e confrontarle con quelle lette in precedenza.

In caso di variazioni il server provvederà a notificare (eventi/allarmi) o salvare nel nodo specifico il valore aggiornato (variabili).

Ogni variabile contenuta nel datamodel sarà salvata su REDIS all'indirizzo

db0:Sour:DataModelNameReplaced

dove DataModelNameReplaced = NS del datamodel (che contiene "/") dove alla backslash si sostituiscono i ":" impiegati da Redis come "folder virtuali.

esempio:

Machine/Axes/AxisX/FeedRate --> Machine:Axes:AxisX:FeedRate

DataModel

Il datamodel necessario ad avviare il server OPC-UA è generato a partire da un file XML, che viene impiegato anche per configurare sia l'adapter (con opportune integrazioni a seconda del tipo di tecnologia) che la struttura dati sul server REDIS che è fatta in modo duale e corrispondente al datamodel stesso.

Questo permette di semplificare debug, gestione e normalizzazione del modello dei dati in quanto lo spazio di memoria del server REDIS diventa l'equivalente di una LUT (LookUpTable) in grado di disaccoppiare il server e il provider informativo a monte.

In questo documento di riferimento per il DataModel sono riportate tutte le informazioni e le specifiche per la redazione del DataModel secondo gli standard SCM implementati dalla soluzione SOUR.

Riferimenti per il server OPC-UA

Per il progetto si è partiti dall'implementazione del server OPC-UA sviluppato per il simulatore MSim impiegato epr testare tutta la IOT platform Maestro Connect di SCM.

In particolare si sono impiegati il serverDecorator (per semplificare l'inclusione del server standard della OPC Foundation ed i relativi metodi) ed i metodi reportEvent e setNodeValue.

E' stato inoltre cambiato il namespace e sono stati man mano eliminati i componenti, le calssi ed i riferimenti che erano relativi al simulatore, al player, ai dati specifici di implementazione su base ESA-KVARA (es modalità conf dei messaggi dai files *.msg).

Modalità di funzionamento SOUR

Il server SOUR effettua i seguenti step:

- Avvio con test funzionamento REDIS server locale (127.0.0.1 / localhost)
- Lettura e scrittura di alcuni parametri base (tra cui le versioni del Server e dell'Adapter) nell'area Sour:Conf
- Lettura del DataModel XML (e salvataggio in locale x avvio server)
- Costruzione della struttura di memoria duale al datamodel (per poter leggere in un colpo solo tutti i valori), lettura della prima versione dei dati in memoria
- Avvio thread periodico di lettura
- Ad ogni esecuzione lettura della memoria, confronto variazione valori 1:1 e processing (variabili --> salvataggio, conditions --> decodifica vettore allarmi attivi tramite vettore messaggi allarme)
- Gestione ulteriore degli allarmi VETO (ovvero silenziati) come elenco su apposita variabile REDIS di quali allarmi NON vanno riportati
- Se avviato in modalità "Test" effettua la creazione delle variabili definite dal datamodel (qualora non presenti).

Dettaglio struttura memoria DB del server REDIS

Di seguito si va a raggruppare per aree di pertinenza i dati scambiati sul server REDIS, a partire dalla folder generale SOUR (tipicamente configurata nel **DB0** di REDIS).

Un ruolo fondamentale è quello del DataModel che è lo schema di riferimento dei dati forniti dal sistema, como descritto nel documento di riferimento per SCM. In questo documento sono riportati nel dettaglio

- il nome dell'elemento
- il nome REDIS per la pubblicazione
- il tipo di dato
- gli elementi ammissibili per il data model
- il livello di "obbligazionarietà" (mandatory/optional)

E ogni datamodel dovrà essere composto a partire dagli elementi e dalle regole indicate.

SOURCE	PATH	Valori	Descrizione
ADAPTER	SOUR	{folder}	folder (logica) principale che contiene i dati per la gestione Server Redis

Adapter - Adp

Dati pubblicati da ADAPTER che indicano la sua versione e stato (per watchdog reciproco con server OPC-UA):

SOURCE	PATH	Valori	Descrizione
ADAPTER	SOUR:Adp	{folder}	Folder che contiene alcune variabili generate dall'Adapter x il check del suo stato
ADAPTER	SOUR:Adp:Heartbeat	UTC datetime ISO 8601	Valore clock che se continua a variare indica che l'adapter è alive e sta caricando dati, nel formato UTC completo **YYYY-MM-DDThh:mm:ss.nnnTZD**, es **2018-11-05T13:15:30.123Z**
ADAPTER	SOUR:Adp:Status	string	Stato dell'adapter (UNKNOWN,started,running,stopped)
ADAPTER	SOUR:Adp:Vers	a.b.c.d	Versione dell'applicativo adapter che carica i dati su redis

Adapter - AdpConf

Dati pubblicati da ADAPTER che indicano la configurazione di riferimento:

SOURCE	PATH	Valori	Descrizione
ADAPTER	SOUR:AdpConf	{folder}	Folder che contiene la configurazione cui fa riferimento l'Adapter
ADAPTER	SOUR:AdpConf:DataModel	XML (string)	File XML che rappresenta il DataModel da implementare da parte del server OPC-UA. Deve essere alimentato dall'**ADAPTER** in modo che il server, all'avvio, lo cerca, salva in locale ed impiega x il setup dei nodi gestiti.
ADAPTER	SOUR:AdpConf:Cnc:Condition:Curr SOUR:AdpConf:Cnc:Condition:It SOUR:AdpConf:Cnc:Condition:En	HashList x 3	vedere specifica comune più sotto **Conditions**
ADAPTER	SOUR:AdpConf:Hmi:Condition:Curr SOUR:AdpConf:Hmi:Condition:It SOUR:AdpConf:Hmi:Condition:En	HashList x 3	vedere specifica comune più sotto **Conditions**
ADAPTER	SOUR:AdpConf:Plc:Condition:Curr SOUR:AdpConf:Plc:Condition:It SOUR:AdpConf:Plc:Condition:En	HashList x 3	vedere specifica comune più sotto **Conditions**

Conditions

- Si tratta di 3 elenchi degli allarmi del CNC (in lingua CURRENT, ITaliano ed ENglish).
- La chiave è una **string** che indica il codice allarme (univoco) anche composto di più parti (codice + parametro) + severity, separate da simbolo hash **|**.
- Il valore è una string di Descrizione (severity es 500=info, 900=fault).
- Può venire popolato a RunTime mano a mano che si manifesta un allarme definito da uno specifico codice univoco

Esempio: **000157000A22|900** → allarme **157**, FAULT (**900**) con parametro **A22**

Adapter - AdpVeto

Dati pubblicati da ADAPTER che indicano la configurazione dei VETO, ovvero gli ALLARMI SILENZIATI (che vengono salvati su REDIS ma che ils erver OPC-UA **NON deve** riportare):

SOURCE	PATH	Valori	Descrizione
ADAPTER	SOUR:AdpVeto	{folder}	Folder che contiene la configurazione dei VETO cui fa riferimento l'Adapter
ADAPTER	SOUR:AdpVeto	{folder}	Cartella che contiene un elenco di VETO relativo agli allarmi "silenziati" che NON VANNO notificati dal server (anche se indicati dall'adapter).
ADAPTER	SOUR:AdpVeto:Cnc:Condition	string	Elenco (comma-separated) di VETO relativo agli allarmi "silenziati" che NON VANNO notificati dal server (anche se indicati dall'adapter).
ADAPTER	SOUR:AdpVeto:Hmi:Condition	string	Elenco (comma-separated) di VETO relativo agli allarmi "silenziati" che NON VANNO notificati dal server (anche se indicati dall'adapter).
ADAPTER	SOUR:AdpVeto:Plc:Condition	string	Elenco (comma-separated) di VETO relativo agli allarmi "silenziati" che NON VANNO notificati dal server (anche se indicati dall'adapter).

Adapter - Machine

Qui si trovano i **veri** dati pubblicati da ADAPTER, ovvero una rappresentazione del datamodel indicato nell'area di AdpConf ove ogni nodo viene valorizzato come di seguito indicato:

SOURCE	PATH	Valori	Descrizione
ADAPTER	SOUR:Machine	{folder}	Cartella principale che contiene l'intero DATAMODEL (specificato in CONF) dove ogni nodo contiene il valore negativo (nodi Variables). In caso di conditions il nodo conterrà un elenco, comma-separated, degli allarmi attivi in un dato momento.
ADAPTER	SOUR:Machine:VariableItem (*SOUR:Machine:Axis:01:AccTime*)	Variable	Valore puntuale di una variabile DataItem: nell'esempio abbiamo indicato il valore che indica le ore cumulate di funzionamento dell'asse 01.
ADAPTER	SOUR:Machine:ConditionItem (*SOUR:Machine:Axis:01:AlarmPLC*)	Condition	Valore puntuale di una una condition specifica: nell'esempio abbiamo indicato il nodo degli allarmi PLC dell'asse 01.

Nota: per i nodi degli allarmi si devono salvare, puntulamente, l'elenco degli allarmi attivi in un dato momento, separati da virgola (es 000012000A320|900, 000252000A120|500) che poi il server SOUR andrà a spacchettare e gestire.

Adapter - Srv

Dati pubblicati da SOUR (SERVER OPC-UA) che indicano versione e stato (per watchdog reciproco con server OPC-UA):

SOURCE	PATH	Valori	Descrizione
SERVER	SOUR:Srv	{folder}	Folder che contiene alcune variabili generate dal SERVER x il check del suo stato
SERVER	SOUR:Srv:Heartbeat	UTC datetime ISO 8601	Valore clock che se continua a variare indica che l'adapter è alive e sta caricando dati, nel formato UTC completo **YYYY-MM-DDThh:mm:ss.nnnTZD**, es **2018-11-05T13:15:30.123Z**
SERVER	SOUR:Srv:Status	string	Stato del SERVER (UNKNOWN,started,running,stopped)
SERVER	SOUR:Srv:StatBitmap	Bitmap (UInt16)	Stato in formato bitmap della catena di comunicazione: da dx a sx SOUR, Client Edge, Internet, MaestroConnect (cloud per macchina); 00001111 = tutto UP, 00000001 va SOLO il server SOUR
SERVER	SOUR:Srv:Vers	string (a.b.c.d)	Versione dell'applicativo SERVER che carica i dati su redis

Adapter - MConnect

Riepilogo valori nell'albero REDIS legati ai canali di comunicazione con il cloud

SOURCE	PATH	Valori	Descrizione
SERVER	SOUR:MConnect	{folder}	Folder che contiene i canali di comunicazione IN/OUT tra HMI e MaestroConnect
SERVER	SOUR:MConnect:ChannelsIN	{folder}	Folder che contiene i canali IN INGRESSO (Maestro Connect --> HMI)
SERVER	SOUR:MConnect:ChannelsIN:DataError:20181105	HashList	Tabella che contiene l'elenco degli errori segnalati
SERVER	SOUR:MConnect:ChannelsIN:AlertHMI	{folder}	Folder che contiene l'elenco degli alert (generati dal cloud) che HMI deve preesentare
SERVER	SOUR:MConnect:ChannelsIN:DataReq	HashList	Tabella richiesta produzione files export verso cloud
SERVER	SOUR:MConnect:ChannelsIN:LogReq	HashList	Tabella che contiene l'elenco degli eventi di notifica verso HMI
SERVER	SOUR:MConnect:ChannelsOUT	{folder}	Folder che contiene i canali IN USCITA (HMI --> Maestro Connect)
SERVER	SOUR:MConnect:ChannelsIN:DataReq	HashList	Tabella files prodotti corrispondenti a richieste
SERVER	SOUR:MConnect:ChannelsOUT:LogReq	HashList	Tabella log prodotti su richiesta

Dettaglio protocollo comunicazione HMI - Maestro Connect

Questa sezione definisce i dati scambiati in modo diretto idealmente tra **HMI** e piattaforma cloud **Maestro Connect**. Anche se le funzionalità qui descritte non dipendono solo dall'ambito di REDIS, si è preferito raccoglierle e descriverle in questo documento per tematica e funzionalità specifica (poiché appunto definiscono l'intero perimetro di IN/OUT tra HMI di MaestroActive e cloud di MaestroConnect). Le specifiche di riferimento sono riportate nel documento condisivo di gruppo, qui si vogliono evidenziare in particolare l'organizzazione e la struttura dei dati sul server REDIS.

In particolare definisce gli aspetti necessari a garantire la realizzazione delle seguenti funzionalità

1. Nuova prestazione di **notifica da parte del cloud di Maestro Connect** di informazioni verso le HMI 2.0 - MaestroConnect può inviare notifiche a HMI 2.0, come ad esempio "Allerta l'operatore HMI 2.0 con opportuna segnalazione visuale per consultare MaestroConnect riguardo l'avvenuta scadenza di un'azione di Manutenzione Ordinaria"
2. Nuova prestazione di **richiesta da MaestroConnect a HMI per eseguire un task** (ad es. produrre un file BackupExpress, file di log eventi HMI, etc). Questa prestazione sarà utilizzata dai ServiceHotliners che potranno così generare e prelevare file di log dalla macchina con comandi "quasi real-time" remoti

Aspetti tecnici salienti

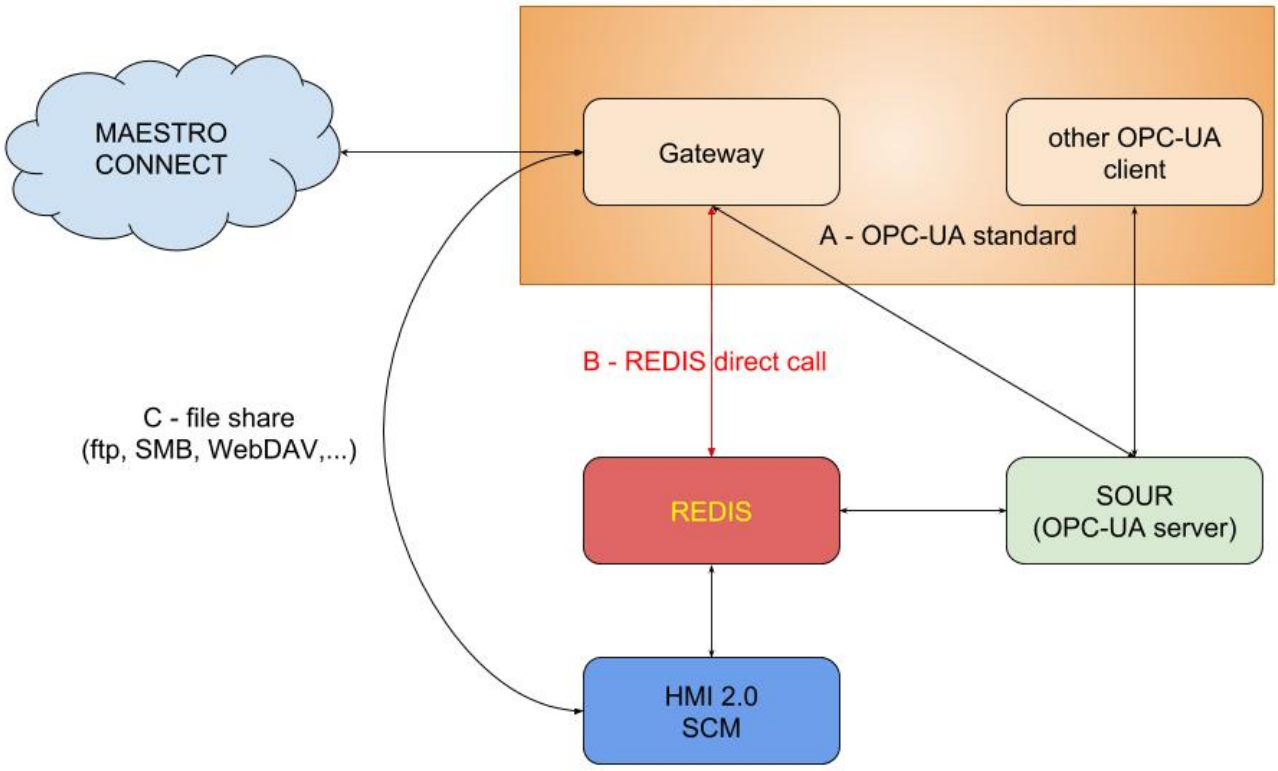
Tutte le prestazioni che si vogliono erogare passano dall'impiego del server REDIS implementato con l'OPC-UA server di gruppo (SOUR) di SCM. In particolare sul server REDIS transiteranno le richieste da e verso il cloud di MaestroConnect.

Per realizzare queste funzionalità sono necessari i seguenti attori

- Cloud Maestro Connect
- Gateway EDGE ONE (con varie docker app tra cui OPC-UA client, ...)
- SOUR - SCM OPC-UA Server
- REDIS memory storage DB
- HMI 2.0

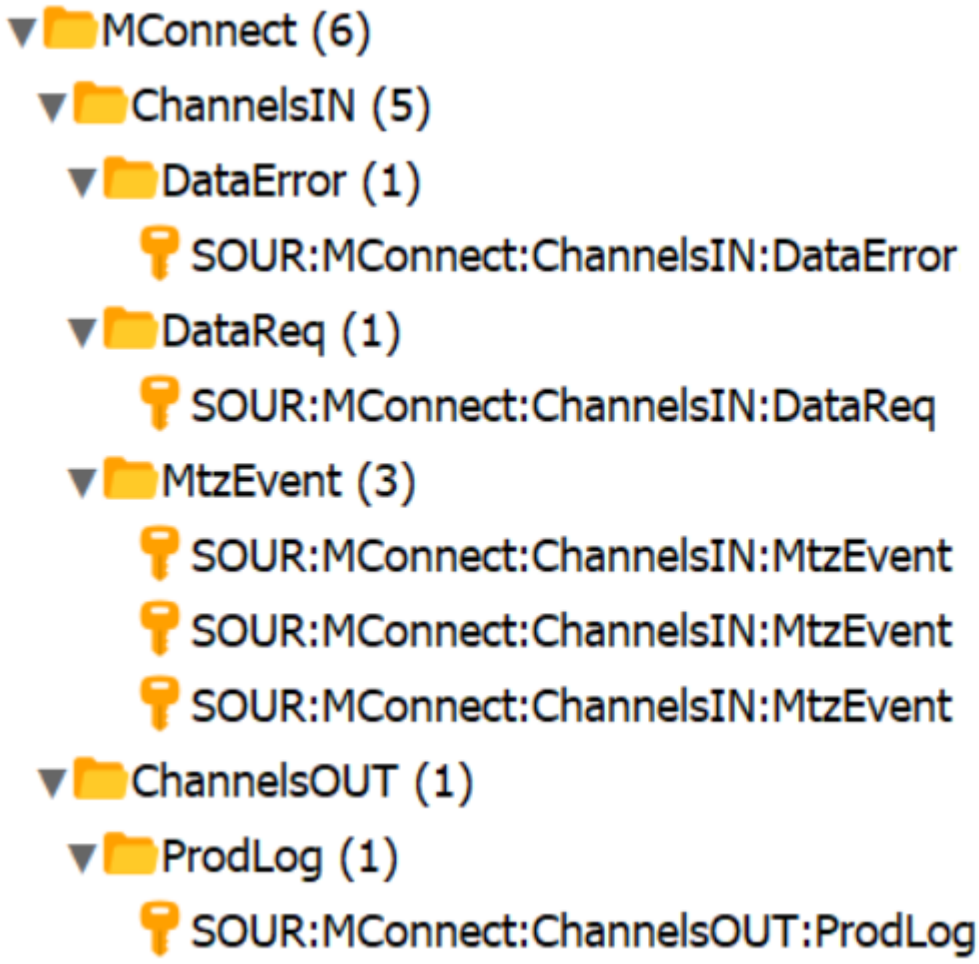
Schema Funzionalità

Di seguito uno schema di massima del sistema di comunicazione



In particolare si ipotizza di impiegare un canale diretto e privilegiato [B] di comunicazione tra Gateway e server REDIS ed un ulteriore canale [C] per il trasferimento di files dall'HMI al Gateway (e da qui poi alla piattaforma cloud).

Si ipotizza di impiegare un'area REDIS per poter svolgere il compito di trasferire informazioni speciali a due vie tra Maestro Connect e HMI 2.0 secondo le modalità di seguito descritte, e con riferimento a questo esempio di partizionamento del DB REDIS:



- Un’area principale **MConnect** come "contenitore logico" dell funzionalità per la connessione diretta, così organizzato:
- **MConnect**
  - **ChannelsIN** (CLOUD → HMI)
  - **ChannelsIN:DataError**: dedicato alla segnalazione degli errori di comunicazione (es variabili dichiarate nel Datamodel ma mancanti) - al momento si tratta di un’area non utilizzata
  - **ChannelsIN:AlertHMI**: dedicato alle notifiche per l'HMI, distinte per tipologia (es **TroubleShooting**, **Maintenance**) - su questo canale vengono generate da MestroConnect le notifiche che devono produrre alert per gli utenti di HMI 2.0. HMI 2.0 le legge, visualizza gli alert, senza cancellarle dal canale REDIS (eventualmente le cancella da interfaccia se l'utente le silenziasse, salvo poi alla successiva verifica riportare di nuovo le notifiche attive).
  - **ChannelsIN:DataReq**: dedicato alle richieste di MaestroConnect per l'esecuzione di un task di produzione dati (per dettagli vedere oltre)
  - **ChannelsIN:LogReq**: dedicato alle richieste per generare i diversi log (di produzione, storico allarmi, eventi) in forma COMPLETA, a fronte di una richiesta da docker app ad HMI riferita ad un preciso intervallo di riferimento.
  - **ChannelsOUT** (HMI → Cloud)
  - **ChannelsOUT:DataReq**: dedicato a registrare l'avvenuta esecuzione dei task richiesti in **ChannelsIN:DataReq**, è il canale in cui l'HMI registra l'avvenuta esecuzione di operazioni di salvataggio file (per dettagli vedere oltre)
  - **ChannelsOUT:LogReq**: dedicato all'output / registrazione dei diversi log di produzione richiesti nel canale duale in ChannelsIN, ovvero storico allarmi, eventi, report di produzione.

- Tutte le informazioni saranno salvate come **HashList** (e quindi come tabelle multiriga) che possono essere popolate in append a partire dal produttore del dato (MaestroConnect per ChannelsIN, HMI per ChannelsOUT).
- Il significato e la gestione dei dati dipenderà dalla tipologia e dalla direzione del canale.

ChannelsIN: notifica info dal Cloud / richiesta esecuzione task -> HMI

In questo caso sarà Maestro Connect che tramite il gateway andrà a scrivere in **REDIS**, nell'area ChannelsIN una informazione destinata all'HMI.

In particolare avremo

- Richiesta all'HMI di presentare un Alert all'utente
- Richiesta all'HMI di produrre un particolare File in Output da salvare in shared folder (gateway)
- Richiesta all'HMI di esportare su REDIS un log di allarmi/eventi/produzione secondo un criterio di finestra temporale richiesta

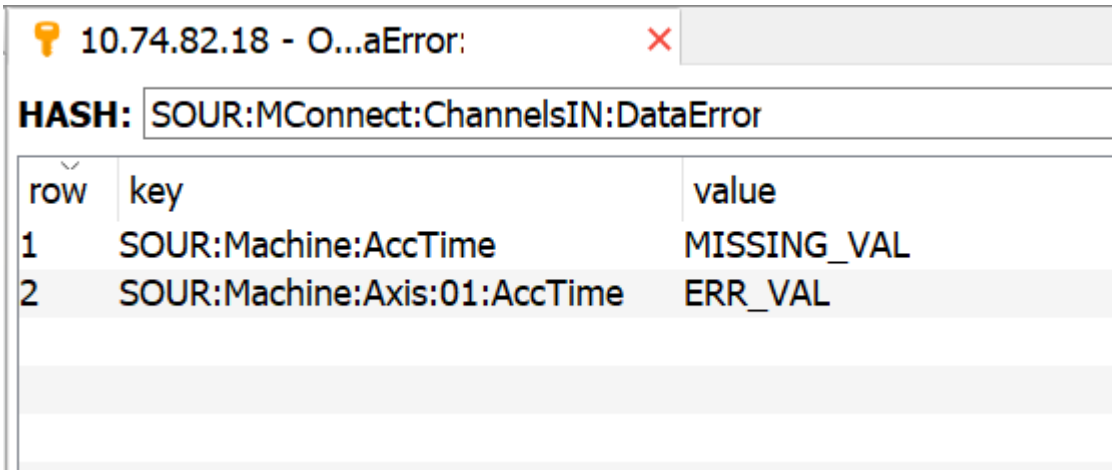
Questa area di REDIS sarà gestita in lettura dall'HMI (che controllerà periodicamente anche con BASSA frequenza nell'arco della giornata) andando a verificare le righe richieste per il contenuto.

Caso d’uso ChannelsIN:DataError

NB: **Attualmente non impiegato**, in attesa di delibera di impiego.

In quest'area si andranno a gestire eventuali errori rilevati dal server SOUR, dal gateway o in generale dal sistema. Ad esempio nella schermata riportata di seguito si rappresentano due errori rilevati dal server SOUR, nel primo caso un campo dichiarato nel DataModel (Machine/AccTime) non è stato trovato da SOUR nell'albero dei dati REDIS e quindi risulta MISSING_VAL.

Esempio:



Caso d’uso ChannelsIN:AlertHMI

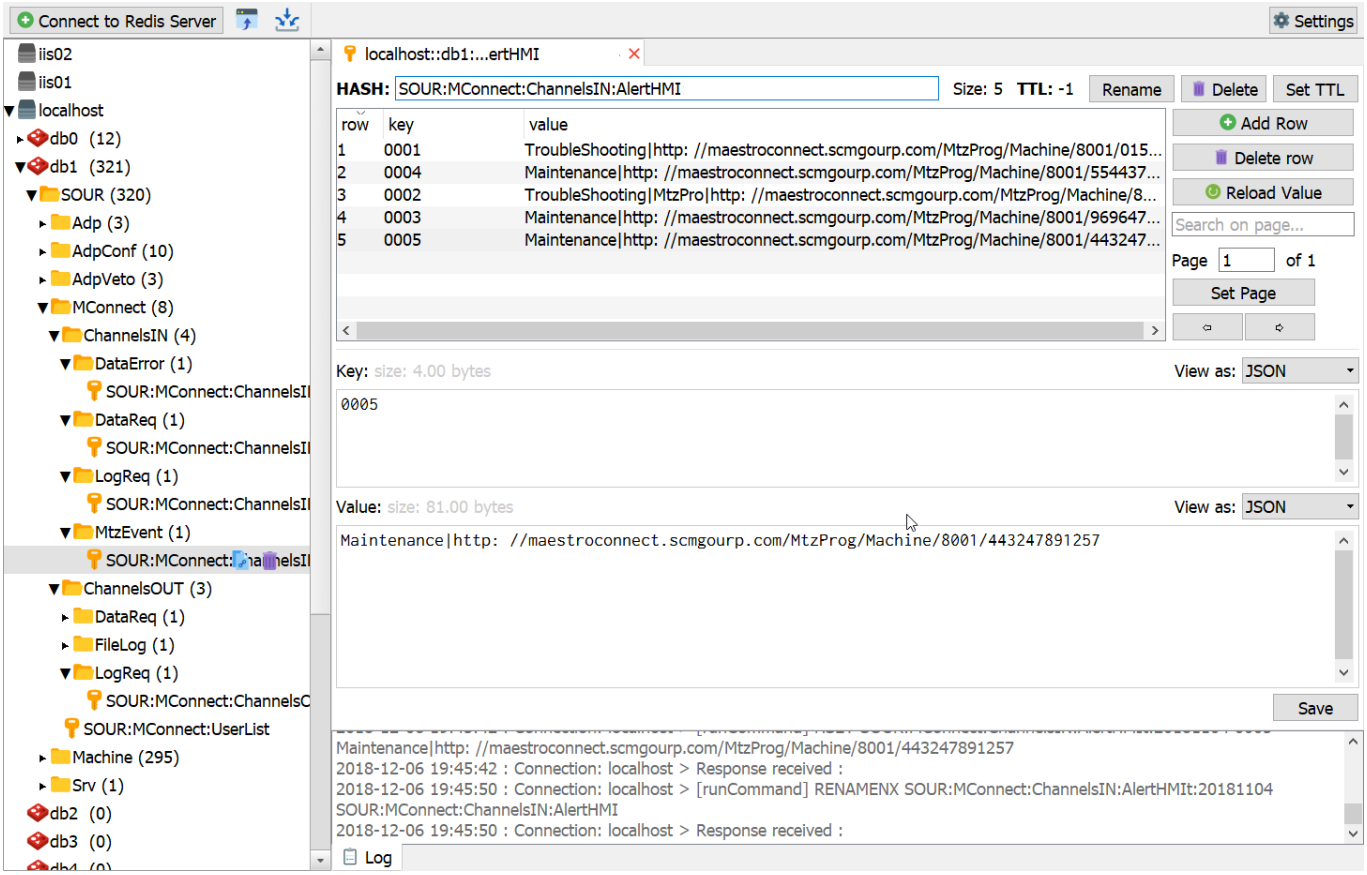
Nel canale **AlertHMI** saranno salvate le richieste di notifica inerenti gli eventi di manutenzione di ogni natura/tipologia, secondo le seguenti regole:

- una HashTable **globale** (che verrà ripulita dal gateway/cloud tramite cancellazione puntuale)
- in ogni HashTable è organizzata in formato Key/Value
- ogni riga ha una chiave UNIVOCA generata dal cloud/gateway
- ogni riga è una notifica indipendente dal resto di un intervento in area Manutenzione
- nella riga il valore contiene tutte le informazioni come stringa con separatore
    - value: **TipoNotifica**|**UrlNotifica**
    - TipoNotifica = **TroubleShooting**, **Maintenance**, eventualmente altri in futuro...
- HMI 2.0 potrà leggere tutte le righe ed eventualmente raggruppare le richieste per tipologia a livello di interfaccia
- L'utente sarà guidato a cliccare sui link per andare su MaestroConnect

In particolare per le 2 tipologie di notifiche ad ora implementate:

- valore "TroubleShooting" - HMI notifica al suo operatore bordo macchina con un colore opportuno un msg, o icona, o qualsivoglia per segnalare "c'è uno o più allarmi da gestire, consulta MaestroConnect"
- valore "Maintenance" - HMI notifica al suo operatore bordo macchina con un colore opportuno diverso dal precedente, per segnalare "c'è una o più azioni di manutenzione ordinaria da eseguire" gli UT decidono poi come gestire un "Cancel" della segnalazione, che ha il solo scopo di "eliminare temporaneamente" la notifica stessa. MaestroConnect cancellerà la URL di notifica SOLO quando un utente autorizzato dichiarerà EFFETTUATA l'azione suggerita per il TroubleShooting o per la Manutenzione La URL inviata da MaestroConnect al momento non genera altre info

Esempio



In questo caso riportato abbiamo

- 3 richieste di manutenzione Ordinaria (**Maintenance**) da manuale macchina ad esempio
- 2 Troubleshooting (**TroubleShooting**) per risolvere un problema minore che si sta verificando

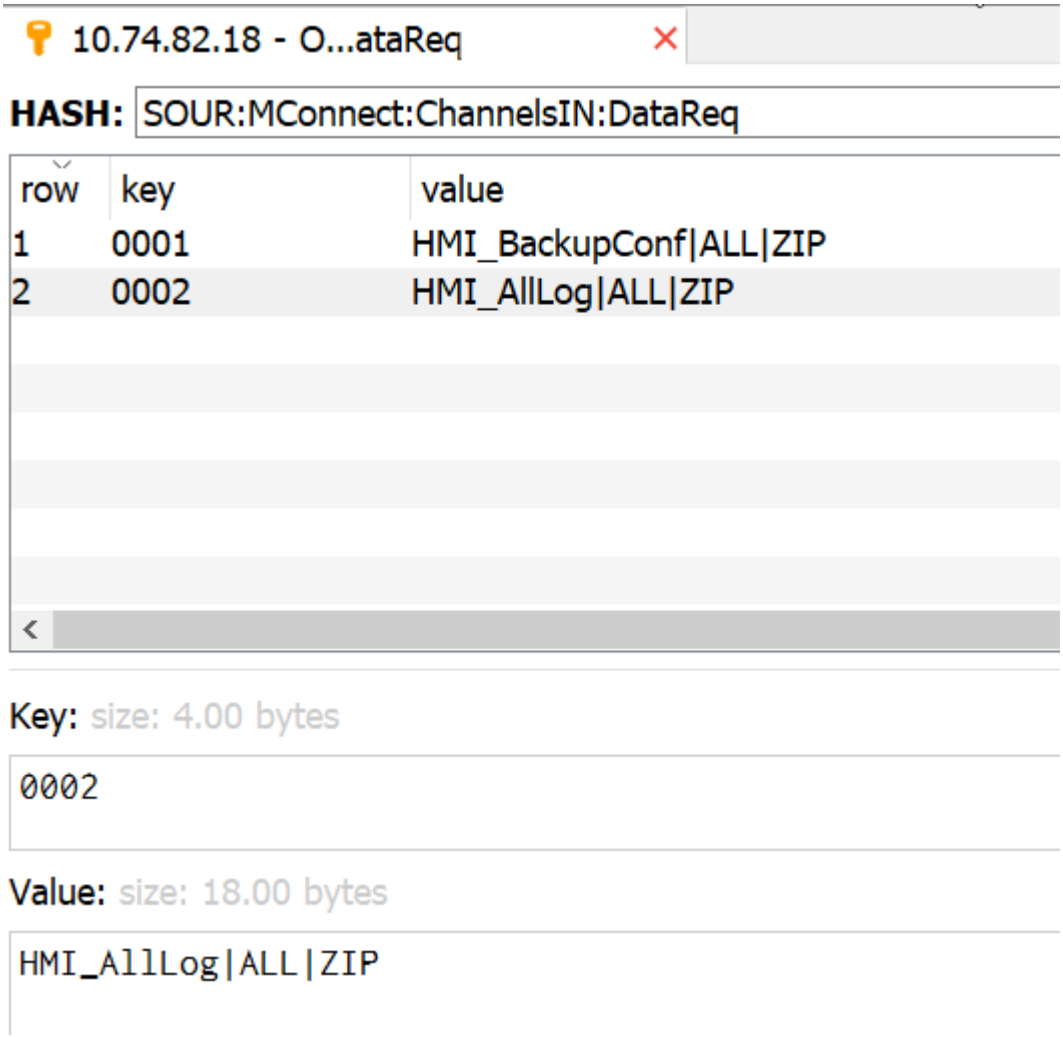
Sarà compito del cloud eliminare tutte le righe di richiesta cessate le cause di insorgenza (ad esempio perché l'operatore va a confermare l'avvenuta esecuzione di un intervento programmato sul cloud di MaestroConnect).

Caso d'uso ChannelsIN:DataReq

Nel canale **DataReq** saranno salvate le richieste di dati, secondo le seguenti regole:

- una HashTable **globale** (che verrà ripulita dal gateway/cloud tramite cancellazione puntuale)
- in ogni HashTable è organizzata in formato Key/Value
- ogni riga ha una chiave UNIVOCA generata dal cloud/gateway come key
- ogni riga è un TASK da eseguire
- nella riga il valore contiene tutti i parametri (specifici per ogni UTE) della richiesta nel formato stringa con separatore (i parametri sono opzionali e a discrezione di ogni UTE)
    - value: **NomeTask|Parametro_01|Parametro_02**
    - il valore di **NomeTask** è liberamente definibile da ogni UTE (es **BackupExpress** per i CDL)
- HMI 2.0 può accedere ad un **folder "esterno", allocato su gateway**, sul quale produce (con opportuno naming) il file richiesto.
- Il nome suggerito per il file sarà nel formato **NomeTask_yyyyMMdd_HHmmss.*** (ovvero con la data e l'ora espresse nel nome)
- Una volta effettuato il salvataggio l'HMI andrà a registrare sul canale duale corrispondente (**ChannelsOUT:DataReq:yyyyMMdd**) l'effettivo salvataggio, registrando una riga con la stessa key della richiesta e con value=nome_file salvato.

Esempio:

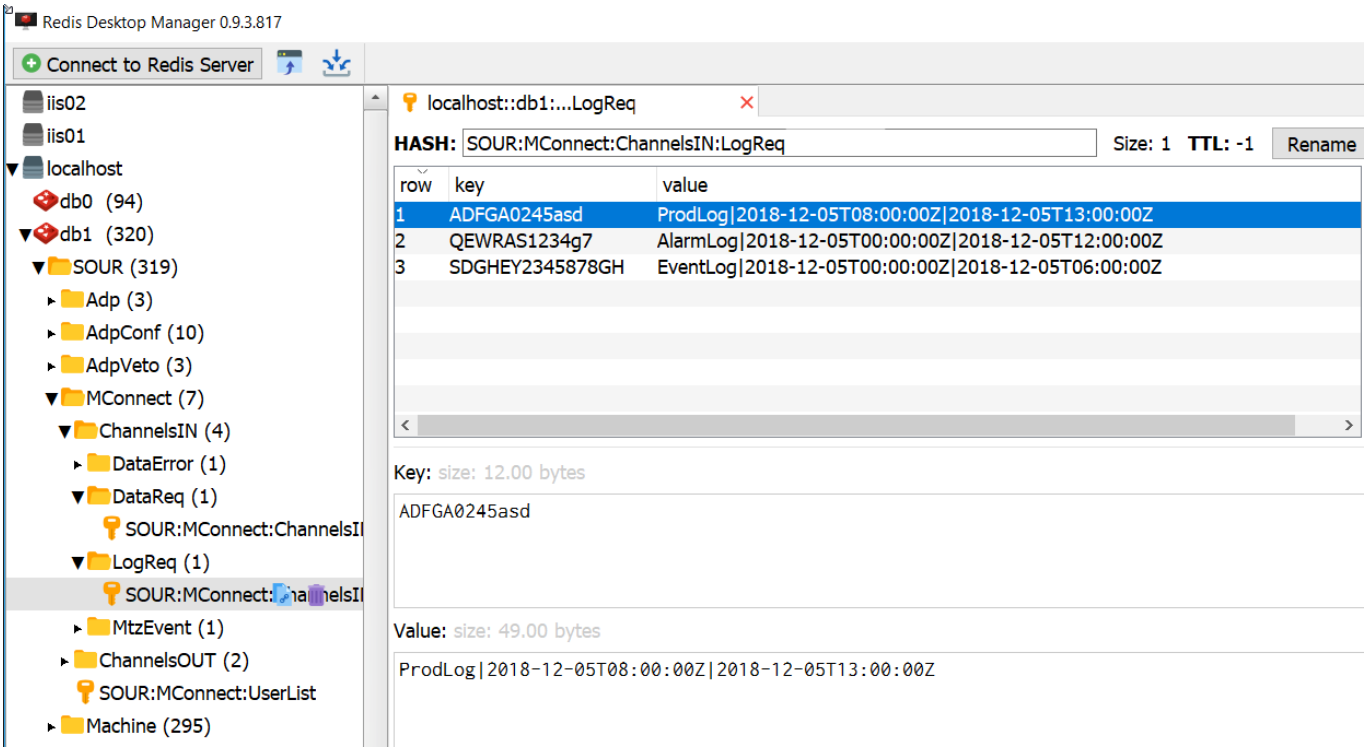


Caso d'uso ChannelsIN:LogReq

Nel canale **LogReq** saranno inviate le richieste di salvataggio dei log secondo le seguenti regole:

- una HashTable **globale** (che verrà ripulita dal gateway/cloud tramite cancellazione puntuale)
- in ogni HashTable è organizzata in formato Key/Value
- ogni riga ha una chiave UNIVOCA generata dal cloud/gateway come key
- ogni riga è un TASK da eseguire
- nella riga il valore contiene tutti i parametri (specifici per ogni UTE) della richiesta nel formato stringa con separatore (i parametri sono opzionali e a discrezione di ogni UTE)
    - value: **TipoDiLog|DataOraInizioPeriodo|DataOraFinePeriodo**
- i tipi di log previsti sono al momento i seguenti (tutti da considerare **opzionali** e da implementare secondo necessità di ogni UTE):
    - **ProdLog**: log di produzione / report di produzione
    - **AlarmLog**: log dello storico degli allarmi
    - **EventLog**: log deglie venti (generico)
- Il formato dataora per le richieste sarà (come per gli altri casi) UTC ISO 8601

Esempio:



Come si vede sono stati registrati 3 richieste:

1. ADFGA0245asd --> ProdLog|2018-12-05T08:00:00Z|2018-12-05T13:00:00Z
    - è stato richiesto il log della produzione, dalle 8 alle 13 UTC
2. QEWRAS1234g7 --> AlarmLog|2018-12-05T00:00:00Z|2018-12-05T12:00:00Z
    - è stato richiesto il log dello storico degli allarmi, dalle 00 alle 12 UTC
3. SDGHEY2345878GH --> EventLog|2018-12-05T00:00:00Z|2018-12-05T06:00:00Z
    - è stato richiesto il log degli eventi, dalle 00 alle 12 UTC

ChannelsOUT: notifica verso Cloud di avvenuta esecuzione di richieste

HMI 2.0 utilizza **una apposita procedura di upload tramite SDK**, tramite la qule i file prodotti da HMI 2.0 sono inviati sul cloud.

La procedura è descritta in maggior dettaglio nella documentazione dedicata all'SDK stesso, ma in termini denerali si tratta di invocare l'SDK come di seguito descritto:

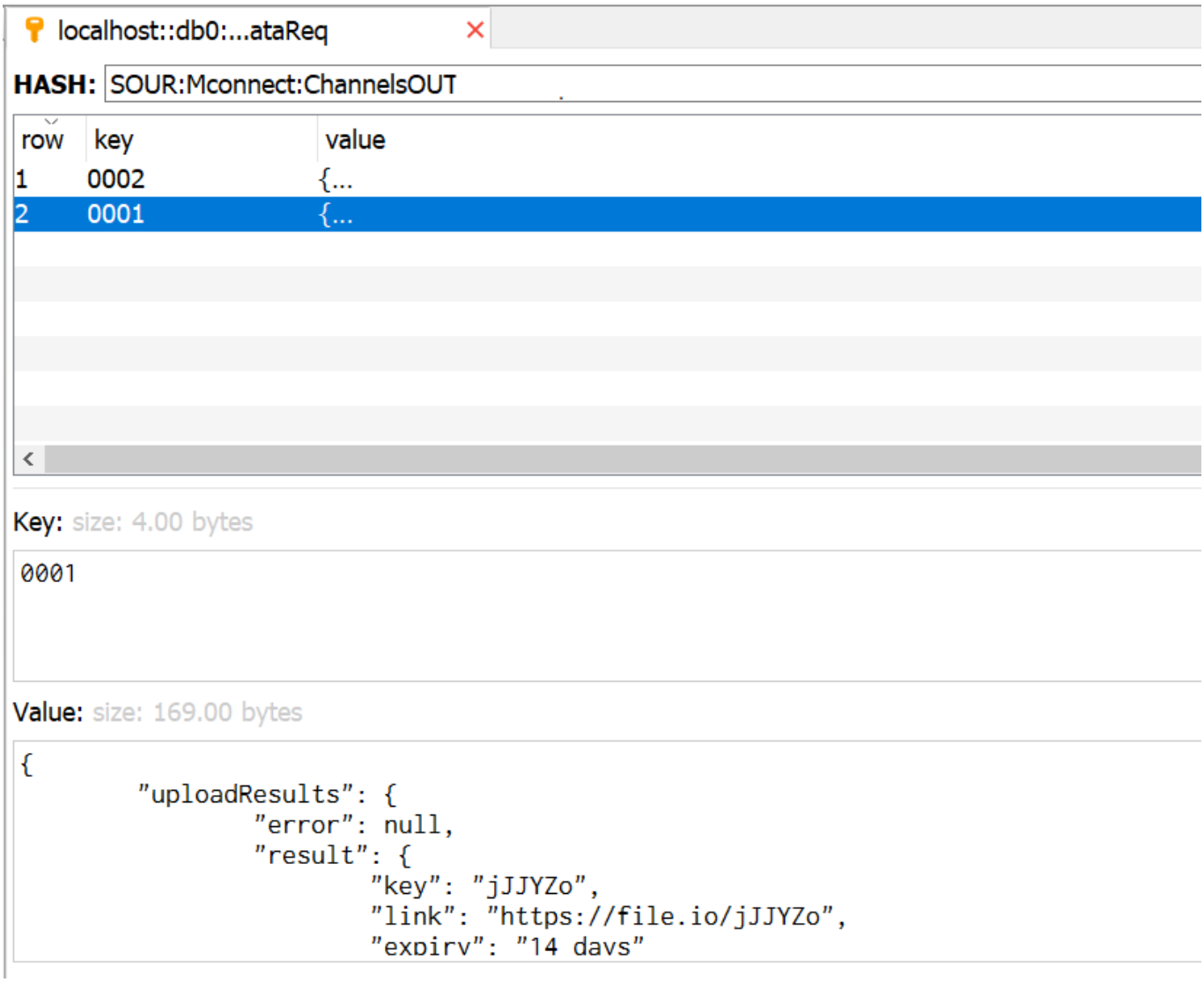
- Chiamata a **TryUpLoadFile**
- Invio di un cancellation token e di un oggetto Progress di tipo Result
- Invio del path completo del file da caricare
- inzio opzionale del nome del file (se si volesse cambiare il nome del file rispetto a qeullo presente sul filesystem, ad esempio per usare un nome standard)

Per agevolare il processo e rendere maggiormente controllata l'esecuzione, si chiede la notifica dei task file-based su apposita area REDIS, copiando il risultato delal chiamata al servizio di pubblicazione direttamente nell'area REDIS di ChannelsOUT corrispondente alla richiesta in ChannelsIN, dove ogni risposta ad un file inviato è relativa ad una richiesta ricevuto tramite ChannelsIN (es: Backupexpress).

Caso d'uso: ChannelsOUT:DataReq

In questo caso sono riportati puntulamente i dati relativi ai richiesti e processati con lo stesso ID della richiesta per poter essere riconciliati.

Esempio:



Esempio JSon **SUCCESSO**

```
{
        "uploadResults": {
                "error": null,
                "result": {
                        "key": "jJJYZo",
                        "link": "https://file.io/jJJYZo",
                        "expiry": "14 days"
                }
        },
        "Message": "Upload Done!"
}
```

Esempio JSon **ERRORE**

```
{
        "uploadResults": {
                "error": {
                        "code": 400,
                        "message": "Trouble uploading file"
                },
                result: null
        },
        "Message": "Upload failed!"
}
```

Come si può notare la richiesta 0002 riportata in precedenza per un log in formato ZIP è stata eseguita ed il file prodotto si può scaricare all'indirizzo https://file.io/jJJYZo.

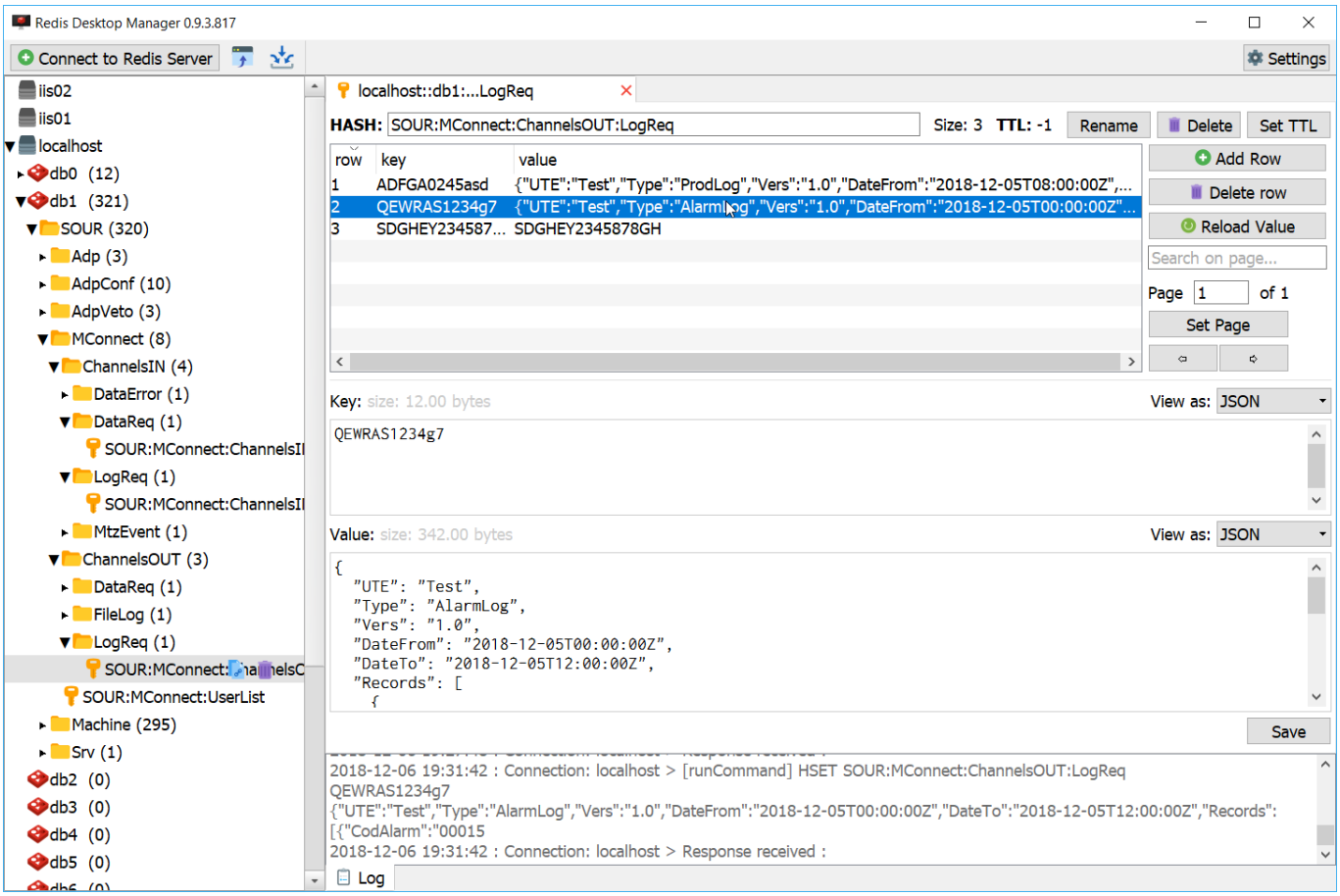
La mancata scrittura in toto della riga indica che il task di produzione del file non è stato ancora preso in carico od eseguito dal sistema, mentre la registrazione di una riga vuota (senza indicazione del nome del file) indica un generico errore in fase di produzione del file per cui sia stato preso in carico, tentata l'elaborazione ma qualcosa ha impedito la produzione o il salvataggio del file come richiesto.

Caso d'uso: ChannelsOUT:LogReq

Nel canale **LogReq** saranno inviate le risposte alle richieste di salvataggio dati di log secondo le seguenti regole:

- una HashTable **globale** (che verrà ripulita dal gateway/cloud tramite cancellazione puntuale)
- in ogni HashTable è organizzata in formato Key/Value
- ogni riga ha una chiave UNIVOCA corrispondente alla richiesta ricevuta (e cui fare riferimento come risposta)
- HMI 2.0 andrà a salvare nel canale duale (ChannelsIN:LogReq) il risultato di ogni richiesta nella forma di un record JSON serializzato corrispondende alla richiesta
- il formato dei dati registrati sarà un oggetto JSON con le seguenti caratteristiche
    - il codice della UTE obbligatorio (ad esempio **BOR/CDL/SEZ**...)
    - tipo di tracciato obbligatorio (tra i valori **ProdLog/AlarmLog/EventLog**)
    - versione del tracciato obbligatoria (in modo che ogni UTE possa implementare una versione del tracciato dati 1.0 con un set di informazioni e possa evolverla in futuro ad un altro set, con differenti informazioni, a patto che il numero di versione permetta alla docker app di trattare correttametne quanto inserito)
    - struttura JSON standard del tipo multi-row enabled, con ogni valore esplicitamente indicato (vedere esempio seguente) comprensiva di qualsiasi valore utile per rappresentare i dati
- data la natura di inefficienza in termini di overhead delle stringhe descrittive del JSON, si da l'indicazione di impiegare questa funzionalità per il recupero di dati giornalieri/settimanali poiché periodi più lunghi potrebbero non essere correttametne gestiti da questa tipologia di scambio dati (in questo caso andrebbe esteso il concetto dei files da inviare tramite DataReq vista nel paragrafo precedente)

Esempio:



Di seguito sono esplicitamente riportati i campi e formati richeisti e concordati nell'incontro hangout del 2019.01.30 per le 3 tipologie di dati, eventualmente declinati secondo le necessità di ogni UTE coinvolta.

AlarmLog

Il log degli allarmi avrà il formato seguente, sono tutti da considerarsi obbligatori tranne le 2 stringhe di traduzione messaggio di errore in lingua IT/EN che possono essere "" (string.empty) e la DateTo che, qualora non fosse valorizzata (string.empty "") significa allarme ancora in corso e non cessato.

Tracciato	Campo	Tipo	Note / Esempio
AlarmLog	SOURCE	STRING	es: PLC/CNC/HMI
AlarmLog	ID	STRING	Identificativo univoco, può essere un unico numero oppure un codice strutturato che comprenda anche eventuali parametri; es 000156, 0012300023400A320
AlarmLog	SEVERITY	INT	Codice severity secondo standard OPC-UA: [0...1000], con 900 = FAULT, 500 = WARNING, 100 = INFO
AlarmLog	Value	Folder/Object	Contenitore messaggi di errore prodotto
AlarmLog	Value/CURR	STRING	Messaggio di errore prodotto, in lingua CORRENTE
AlarmLog	Value/EN	STRING	Messaggio di errore prodotto, in lingua ENGLISH (opzionale)
AlarmLog	Value/IT	STRING	Messaggio di errore prodotto, in lingua ITALIANO (opzionale)
AlarmLog	DateFrom	DATETIME ISO 8601	DataOra dell'inizio/insorgenza della condizione di allarme
AlarmLog	DateTo	DATETIME ISO 8601	DataOra di cessazione della condizione di allarme, SE VUOTO (string.empty) significa allarme ancora attivo
AlarmLog	User	STRING	Codice utente loggato su macchina; NB: NON significa che sia stato riportato o riconosciuto dall'utente

Esempio di tracciato allarme, generato sulla base di una richiesta del log degli allarmi indicata in precedenza, QEWRAS1234g7 --> {...} il log dello storico degli allarmi, dalle 00 alle 12 UTC

NB: un allarme ancora attivo al momento della produzione del log, un altro iniziato e cessato.


```
{
  "UTE": "CDL",
  "Type": "AlarmLog",
  "DateFrom": "2018-12-05T00:00:00.000Z",
  "DateTo": "2018-12-05T12:00:00.000Z",
  "Vers": "1.0",
  "Records": [
    {
      "Source": "CNC",
      "CodAlarm": "1061",
      "Severity": 900,
      "Value": {
        "CURR": "CN1061: X:Allarme Asse: Richiesta Esterna",
        "en-EN": "CN1061: Alarm X Axis: External Request",
        "it-IT": "CN1061: X:Allarme Asse: Richiesta Esterna"
      },
      "DateFrom": "2018-12-05T11:33:33.475Z",
      "DateTo": "",
      "User": "admin"
    },
    {
      "Source": "CNC",
      "CodAlarm": "1065",
      "Severity": 900,
      "Value": {
        "CURR": "CN1061: X:Allarme Asse: Fuoricorsa",
        "en-EN": "CN1065: Alarm X Axis: Off-axis",
        "it-IT": "CN1065: X:Allarme Asse: Fuoricorsa"
      },
      "DateFrom": "2018-12-05T10:33:33.475Z",
      "DateTo": "2018-12-05T10:35:10.731Z",
      "User": "admin"
    }
  ]
}
```

EventLog

Il log degli eventi avrà il formato seguente, sono tutti da considerarsi obbligatori tranne le stringhe di traduzione messaggio dell'evento che sono TUTTE opzionalmente valorizzabili, altrimenti "" (string.empty). Ogni UTE potrà autonomamente decidere COSA costituisca un evento che meriti di venire registrato.

Tracciato	Campo	Tipo	Note / Esempio
EventLog	Type	STRING	Tipologia evento (per codifica secondo encessità UTE)
EventLog	Value	Folder/Object	Contenitore messaggi di errore prodotto
EventLog	Value/CURR	STRING	Messaggio di evento registrato, in lingua CORRENTE
EventLog	Value/EN	STRING	Messaggio di evento registrato, in lingua ENGLISH (opzionale)
EventLog	Value/IT	STRING	Messaggio di evento registrato, in lingua ITALIANO (opzionale)
EventLog	DateFrom	DATETIME ISO 8601	DataOra dell'inizio/insorgenza dell'EVENTO REGISTRATO
EventLog	User	STRING	Codice utente loggato su macchina; NB: NON significa che sia stato riportato o riconosciuto dall'utente

Esempio di tracciato evento, generato sulla base di una richiesta del log degli eventi indicata in precedenza, SDGHEY2345878GH --> {...} il log degli eventi, dalle 00 alle 12 UTC

NB: un evento ha associata una descrizione solo in lingua corrente, il secondo un parametro in OGNI lingua, mentre l'ultimo nessuna descrizione/parametro associato.


```
{
  "UTE": "CDL",
  "Type": "EventLog",
  "DateFrom": "2018-12-05T00:00:00.000Z",
  "DateTo": "2018-12-05T12:00:00.000Z",
  "Vers": "1.0",
  "Records": [
    {
      "Type": "report_application_started_event_type_name",
      "Value": {
        "CURR": "X0 Y-500 T6",
        "en-EN": "",
        "it-IT": ""
      },
      "DateFrom": "2018-12-05T08:33:33.475Z",
      "User": ""
    },
    {
      "Type": "report_application_started_event_type_name",
      "Value": {
        "CURR": "X0 Y-800 T6",
        "en-EN": "X0 Y-800 T6",
        "it-IT": "X0 Y-800 T6"
      }
      "DateFrom": "2018-12-05T09:33:33.475Z",
      "User": ""
    },
    {
      "Type": "report_application_started_event_type_name",
      "Value": {
        "CURR": "",
        "en-EN": "",
        "it-IT": ""
      }
      "DateFrom": "2018-12-05T10:33:33.475Z",
      "User": ""
    }
  ]
}
```

ProductionLog - generale

Il log della produzione sarà ovviamente diverso e specifico per ogni UTE. Inoltre i campi da valorizzare saranno spesso opzionali ed esplicitamente indicati (ove non disponibili sarà indicata stringa vuota "" string.empty), vedere colonna M/O dove M = Mandatory, O = optional

Per questo motivo si vanno a dettagliare i singoli casi delle singole UTE

ProductionLog - CDL e Foratrici

- Sarà prodotto 1 record per ogni singolo pezzo prodotto = singola esecuzione del programma. In caso di N-ripetizioni del medesimo programma saranno prodotti N record
- M/O = Mandatory / Optional

Tracciato	Campo	Tipo	M/O	Note / Esempio
ProdLog/CDL	Name	STRING	M	Nome del programma in esecuzione
ProdLog/CDL	DimensionX	FLOAT	M	Dimensione lavorazione X
ProdLog/CDL	DimensionY	FLOAT	M	Dimensione lavorazione Y
ProdLog/CDL	DimensionZ	FLOAT	O	Dimensione lavorazione Z (opzionale)
ProdLog/CDL	DateFrom	DATETIME ISO 8601	M	DataOra dell'inizio esecuzione del programma / produzione
ProdLog/CDL	DateTo	DATETIME ISO 8601	M	DataOra della fine esecuzione del programma / produzione
ProdLog/CDL	ExpectedDuration	INT	O	Durata attesa in secondi di esecuzione, se = 0 --> non disponibile; calcolato solo per .PGMX
ProdLog/CDL	IsWaste	INT	M	Codifica alfanumerica della qualità del pezzo: 0 = buono, 1 = scarto (eventuali altri valori futuri)
ProdLog/CDL	User	STRING	M	Codice utente loggato su macchina; NB: NON significa che sia stato riportato o riconosciuto dall'utente
ProdLog/CDL	WorkArea	STRING	O	Area di lavoro, opzionale e mai valorizzato per foratrici

Esempio di tracciato evento, generato sulla base di una richiesta del log degli eventi indicata in precedenza, ADFGA0245asd --> {...} il log della produzione, dalle 8 alle 13 UTC


```
{
  "UTE": "CDL",
  "Type": "ProdLog",
  "DateFrom": "2018-12-05T08:00:00.000Z",
  "DateTo": "2018-12-05T13:00:00.000Z",
  "Vers": "1.0",
  "Records": [
     {
      "Name": "riccardo.pgmx",
      "DimensionX": 1600.0,
      "DimensionY": 1200.0,
      "DimensionZ": 50.0,
      "DateFrom": "2018-12-05T11:48:52.073Z",
      "DateTo": "2018-12-05T11:49:48.455Z",
      "ExpectedDuration": 21,
      "IsWaste": 0,
      "User": "admin",
      "Workarea": "AB"
    },
    {
      "Name": "riccardo.pgmx",
      "DimensionX": 1600.0,
      "DimensionY": 1200.0,
      "DimensionZ": 50.0,
      "DateFrom": "2018-12-05T11:54:49.472Z",
      "DateTo": "2018-12-05T11:59:31.297Z",
      "ExpectedDuration": 21,
      "IsWaste": 0,
      "User": "admin",
      "Workarea": "AB"
    }
  ]
}
```

ProductionLog - Sezionatrici

- E' necessario un ulteriore oggetto strutturato per rappresentare la composizione di ogni MIX come tipo di sagome e quantità
- M/O = Mandatory / Optional

Tracciato	Campo	Tipo	M/O	Note / Esempio
ProdLog/SEZ	ODP	STRING	M	Ordine di Produzione
ProdLog/SEZ	Name	STRING	M	Nome del programma in esecuzione
ProdLog/SEZ	InfoMIX	ARRAY [**ShapeSEZ**]	M	Array che rappresenta le sagome tagliate
ProdLog/SEZ	DimensionX	FLOAT	M	Dimensione lavorazione X = lunghezza
ProdLog/SEZ	DimensionY	FLOAT	M	Dimensione lavorazione Y = larghezza
ProdLog/SEZ	DimensionZ	FLOAT	O	Dimensione lavorazione Z = spessore (uguale per il gruppo di sagome)
ProdLog/SEZ	MatType	STRING	M	Stringa x codifica TIPOLOGIA materiale, lingua corrente
ProdLog/SEZ	MatForm	STRING	M	Stringa x codifica FORMATO materiale, lingua corrente
ProdLog/SEZ	MatQty	INT	M	Quantità di materiale impiegato (secondo tipo e formato) sopra definiti
ProdLog/SEZ	DateFrom	DATETIME ISO 8601	M	DataOra dell'inizio esecuzione del programma / produzione
ProdLog/SEZ	DateTo	DATETIME ISO 8601	M	DataOra della fine esecuzione del programma / produzione
ProdLog/SEZ	ExpectedDuration	INT	O	Durata attesa in secondi di esecuzione, se = 0 --> non disponibile
ProdLog/SEZ	RealDuration	INT	O	Durata effettiva in secondi di esecuzione, se = 0 --> non disponibile
ProdLog/SEZ	User	STRING	M	Codice utente loggato su macchina; NB: NON significa che sia stato riportato o riconosciuto dall'utente

Dettaglio InfoMIX = array di **ShapeSEZ**

Tracciato	Campo	Tipo	M/O	Note / Esempio
ShapeSEZ	Code	STRING	M	Codice identificativo della sagoma

Tracciato	Campo	Tipo	M/O	Note / Esempio
ShapeSEZ	DimensionX	FLOAT	M	Dimensione lavorazione X = lunghezza
ShapeSEZ	DimensionY	FLOAT	M	Dimensione lavorazione Y = larghezza
ShapeSEZ	DimensionZ	FLOAT	O	Dimensione lavorazione Z = spessore
ShapeSEZ	Qty	INT	M	Quantità di sagome presenti da tagliare

Esempio di tracciato evento, generato sulla base di una richiesta del log degli eventi indicata in precedenza, ADFGA0245asd --> {...} il log della produzione, dalle 8 alle 13 UTC


```
{
  "UTE": "SEZ",
  "Type": "ProdLog",
  "DateFrom": "2018-12-05T08:00:00.000Z",
  "DateTo": "2018-12-05T13:00:00.000Z",
  "Vers": "1.0",
  "Records": [
    {
      "ODP": "PRIMO.MIX",
      "Name": "Taglio002.prg",
      "InfoMIX": [
        {
          "Code": "alfa",
          "DimensionX": 1200.0,
          "DimensionY": 800.0,
          "DimensionZ": 50.0,
          "Qty": 5
        },
        {
          "Code": "beta",
          "DimensionX": 1000.0,
          "DimensionY": 500.0,
          "DimensionZ": 50.0,
          "Qty": 7
        }
      ],
      "DimensionX": 1600.0,
      "DimensionY": 1200.0,
      "DimensionZ": 50.0,
      "MatType": "MDF_0012",
      "MatForm": "Fogli",
      "MatQty": 5,
      "DateFrom": "2018-10-04T11:48:52.073Z",
      "DateTo": "2018-10-04T11:49:52.073Z",
      "ExpectedDuration": 65,
      "RealDuration": 60,
      "User": "admin"
    },
    {
      "ODP": "PRIMO.MIX",
      "Name": "Taglio005.prg",
      "InfoMIX": [
        {
          "Code": "gamma",
          "DimensionX": 1200.0,
          "DimensionY": 800.0,
          "DimensionZ": 50.0,
          "Qty": 4
        },
```

```
        {
          "Code": "delta",
          "DimensionX": 1000.0,
          "DimensionY": 500.0,
          "DimensionZ": 50.0,
          "Qty": 8
        }
      ],
      "DimensionX": 1600.0,
      "DimensionY": 1200.0,
      "DimensionZ": 50.0,
      "MatType": "MDF_0013",
      "MatForm": "Fogli",
      "MatQty": 5,
      "DateFrom": "2018-10-04T11:49:59.073Z",
      "DateTo": "2018-10-04T11:50:59.073Z",
      "ExpectedDuration": 65,
      "RealDuration": 60,
      "User": "admin"
    }
  ]
}
```

ProductionLog - Bordatrici

Regole specifiche Bordatrici:

- Sarà prodotto 1 record per ogni singolo pezzo lavorato = singola esecuzione del programma. In caso di N-ripetizioni del medesimo programma saranno prodotti N record
- M/O = Mandatory / Optional
- nella maggior parte dei casi la larghezza della lavorazione è fissa e/o ininfluente

Tracciato	Campo	Tipo	M/O	Note / Esempio
ProdLog/BOR	Name	STRING	M	Nome del programma in esecuzione
ProdLog/BOR	DimensionX	FLOAT	M	Dimensione lavorazione X = lunghezza
ProdLog/BOR	DimensionY	FLOAT	O	Dimensione lavorazione Y = larghezza (per molte tipologie è fissa, in tal caso ammesso 0)
ProdLog/BOR	DimensionZ	FLOAT	M	Dimensione lavorazione Z = spessore (uguale per il gruppo di sagome)
ProdLog/BOR	Pass	INT	M	Indicazione passaggio [1,2]
ProdLog/BOR	BordCode	STRING	M	Stringa x codifica del tipo di bordo (user defined), lingua corrente
ProdLog/BOR	BordCons	FLOAT	M	Quantità di bordo consumato per la lavorazione
ProdLog/BOR	DateFrom	DATETIME ISO 8601	M	DataOra dell'inizio esecuzione del programma / produzione
ProdLog/BOR	User	STRING	M	Codice utente loggato su macchina; NB: NON significa che sia stato riportato o riconosciuto dall'utente

Esempio di tracciato evento, generato sulla base di una richiesta del log degli eventi indicata in precedenza, ADFGA0245asd --> {...} il log della produzione, dalle 8 alle 13 UTC


```
{
  "UTE": "BOR",
  "Type": "ProdLog",
  "DateFrom": "2018-12-05T08:00:00.000Z",
  "DateTo": "2018-12-05T13:00:00.000Z",
  "Vers": "1.0",
  "Records": [
    {
      "Name": "Bordo00231.prg",
      "DimensionX": 1600.0,
      "DimensionY": 1200.0,
      "DimensionZ": 20.0,
      "Pass": 1,
      "BordCode": "MELAN_01",
      "BordCons": 1620.0,
      "DateFrom": "2018-10-04T11:48:52.073Z",
      "User": "admin"
    },
    {
      "Name": "Bordo00231.prg",
      "DimensionX": 1200.0,
      "DimensionY": 1200.0,
      "DimensionZ": 20.0,
      "Pass": 1,
      "BordCode": "MELAN_01",
      "BordCons": 1220.0,
      "DateFrom": "2018-10-04T11:48:54.073Z",
      "User": "admin"
    }
  ]
}
```