UNIVERSITÀ DEGLI STUDI DI CATANIA

OPC-UA Aggregation Server

Industrial Informatics a.a 2019/2020

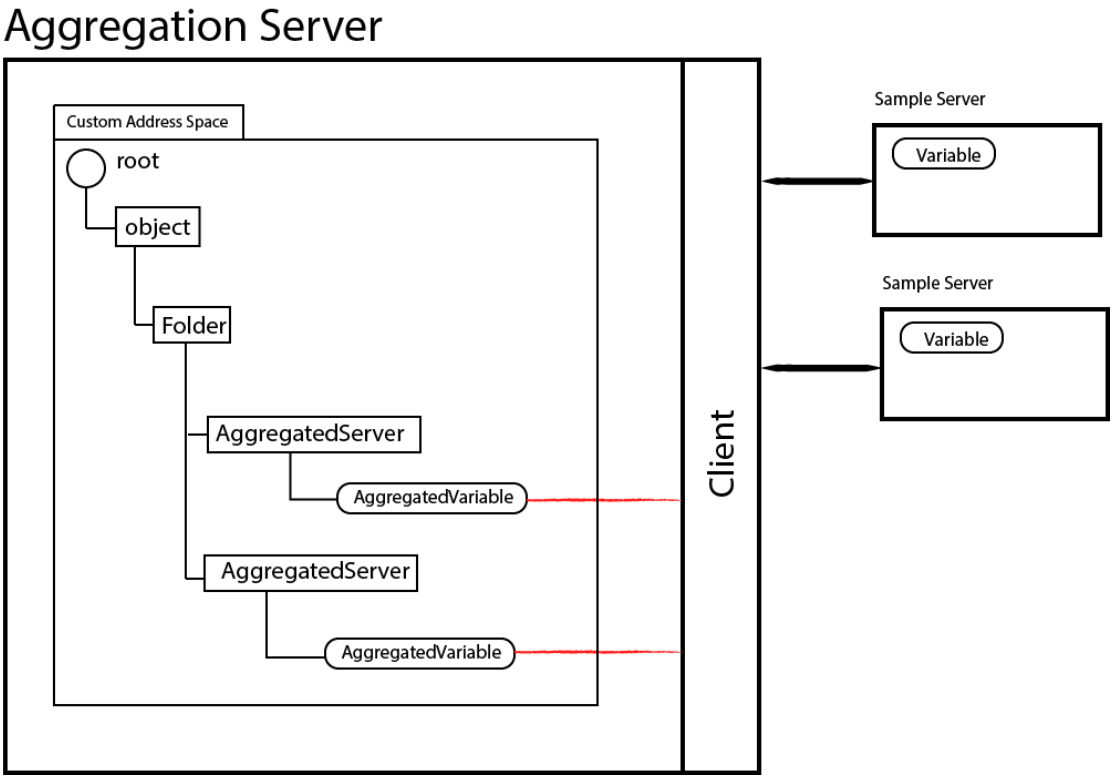
Raiti Mario O55000434
Nardo Gabriele Salvatore O55000430

Lo scopo di questa tesina di fine corso è la realizzazione di un Aggregation Server utilizzando la versione in python dello stack OPC-UA , disponibile gratuitamente su github al seguente link (https://github.com/FreeOpcUa/python-opcua).

Il codice sorgente dell'elaborato è disponibile su github al corrispondente indirizzo (aggiungere link).

Sommario

Aggregation Server – Architettura



La figura in alto mostra l'architettura di base dell'elaborato. L'elemento Aggregation server sarà un Server OPC-UA. L'address space è stato customizzato creando un nuovo namespace specifico per l'applicazione e ai suoi componenti di base è stato aggiunto un Node di tipo folder che avrà lo scopo di raccogliere e organizzare gli oggetti AggregatedServer. Tali oggetti modellano i sample server che verranno aggregati , a tale proposito è stato creato un nuovo Object Type custom chiamato proprio AggratedServer a cui è stato aggiunto un set di variabili che modellano i valori di cui si vuole tener traccia.

All'interno dell'aggregation server è previsto un modulo client che avrà il compito di stabilire le connessioni con i sample server al fine di leggere e scrivere le variabili di cui si vuole tenere traccia. Le informazioni relative ai sample server da aggregare , e che quindi il modulo client deve raggiungere , sono contenuti all'interno di un file di configurazione in formato *json* (tale file verrà discusso in dettaglio in seguito) in cui sono anche indicati i nodeId delle informazioni da recuperare e le modalità di recupero cioè tramite subscription o polling (read/write).

I valori prelevati dal modulo client devono essere sincronizzati con le copie locali dell'aggregation server cioè le variable degli AggregatedServer , per tale scopo tali dati devono mantenere come source timestamp quello del sample server.

File di Configurazione

In questa sezione verranno descritti i file di configurazione json utilizzati per il passaggio delle informazioni di configurazione e per la creazione dei certificati x509v3.

Config.json

Questo file contiene un elemento sample server per ogni server che si vuole aggregare, per ogni elemento sono previsti 6 capi da configurare opportunamente per settare le informazioni relative al servere e ai dati da tracciare. Di seguito vengono descritti tali campi, per ognuno di essi sarà presentato in basso un esempio di valore e la possibilità dei valori ammissibili :

- **Endpoint** : deve contenere l'url del server che si vuole aggregare,
- **security_policy** : deve contenere una stringa che rappresenti l'algoritmo utilizzato per le operazioni di sicurezza ove previste , in accordo al campo security mode,
- **security_mode** : deve contenere una stringa contente la modalità di sicurezza richiesta , i valori ammissibili sono None , Sign e SignAndEncrypt,
- **node_id** : deve contenere il node id della variabile di cui si vogliono ottenere i valori soto forma di stringa formattata nel seguente modo ns=valore;i=valore,
- **variable_type** : deve contenere il tipo della variabile da leggere,
- **service_req** : definisce il tipo di servizio per ottenere i dati , i valori ammissibili sono due , *polling* per abilitare un accesso ai dati utilizzando i servizi read/write e *subscribe* per abilitare l'accesso ai dati in modalità pub/sub
- **publish_interval** : da settare solo se si sceglie come valore di service_req *subscribe*, inserire un intero senza segno , tenere conto che è il valore indicherà millisecondi.

In basso viene riportato un esempio di come riempire i campi del suddetto file.

```
{
   "sample_server1" : {
        "endpoint":"opc.tcp://pc-mario:51210/UA/SampleServer",
        "security_policy":"None",
        "security_mode":"None",
        "node_id":"ns=2;i=10852",
        "variable_type":"DataValue",
        "service_req":"polling",
        "publish_interval": ""
    },
}
```

Openssl_conf.json

In questo file sono presenti due campi da settare opportunamente per la corretta esecuzione del programma , utilizzati per la creazione dei certificati. Anche questo file è in formato *json* è composto da due campi :

- **ssl_installation_path** in questo campo va inserito il proprio path di installazione di openssl.
- **ssl_confing_file_name** va inserito il nome del file di configurazione di openssl.

In basso viene riportato un esempio di come riempire i campi del suddetto file.

```
{
    "ssl_installation_path":"C:\Program Files\OpenSSL-Win64\bin\openssl.cfg",
    "ssl_confing_file_name":"openssl.cfg"
}
```

Implementazione

In questa sezione saranno mostrati i dettagli implementativi , discusse le scelte progettuali e le funzionalità sviluppate nei file *aggregationServer.py* e *Client.py*.

L'elaborato è stato sviluppato in ambiente Windows (Windows 10 Professional), per tale motivo le scelte implementative sono mirate all'esecuzione su tale piattaforma (gestione dei path), da ciò ne consegue che potrebbero incorrere errori durante l'esecuzione su altre piattaforme diverse da quella presa in considerazione.

Il codice è stato sviluppato utilizzando l'ultima versione di python cioè la 3.8. Come editor è stato utilizzato *VScode*.

Aggregation Server

(Descrivere in maniera dettagliata il file aggragtionServer.py)

Client

(Descrivere in maniera dettagliata il file Client.py)

Risultati

(Qui potremmo discutere i risulatati delle esecuzioni con screen)

Altri Dettagli

I dettagli relativi alla struttura del progetto e all'avvio dell'applicativo sono contenuti all'interno del file ***README*** contenuto nella repository del progetto linkata in alto.