Corso di laurea magistrale in ingegneria informatica

Corso di Industrial Informatics

Client OPC UA in Javascript su raspberry

Studenti

Coppolino Antonino

Labruna Mauro

Professore

Salvatore Cavalieri

Indice

Introduzione

Nel seguito del documento sarà spiegato step-by-step il processo di implementazione di un Client OPC-UA scritto in javascript, tale client girerà su raspberry. Le funzionalità sviluppate sono:

- Create session
- Read
- Write
- Browsing
- Create a subscription
- Delete an existing subscription
- Create monitored item

Strumenti utilizzati

- Visual studio code
- Rasbperry PI 3
- OPC Sample Server (OPC Fundation)
- node-opcua
- inquirer https://www.npmjs.com/package/inquirer

Avvio client

Per prima cosa è necessario comunicare con raspberry via ssh, nel nostro caso specifico, lavorando su windows per instaurare la comunicazione ci siamo avvalsi del tool PuTTY.

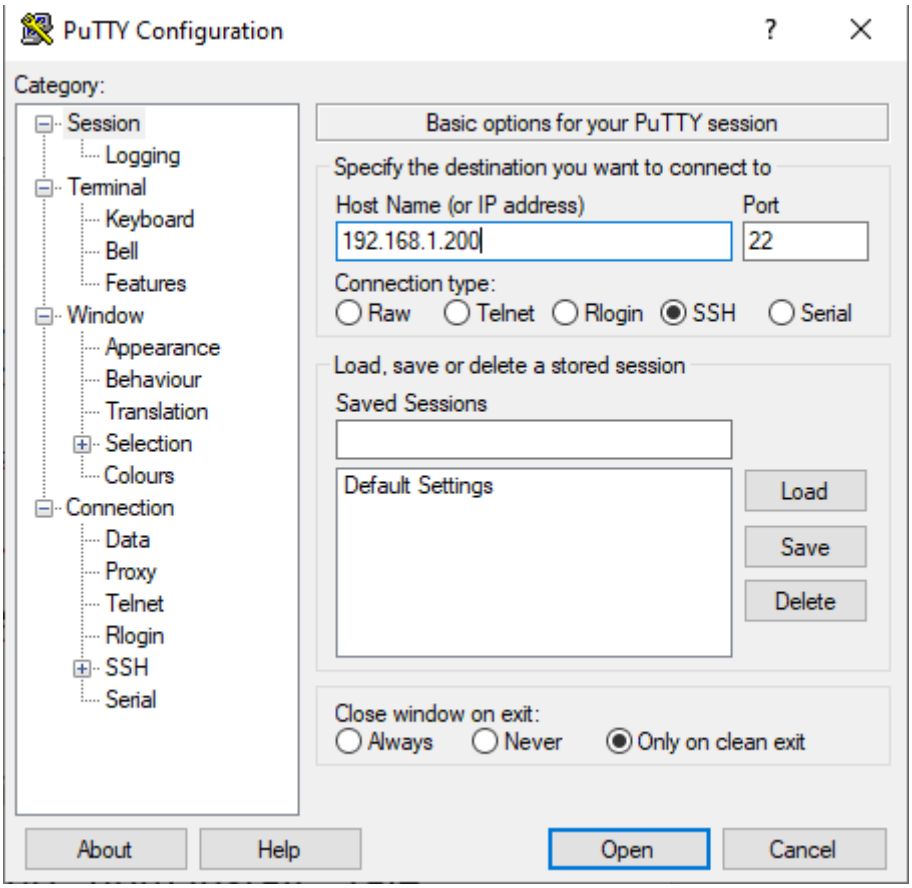

Figura 1 PuTTY Configuration

Come mostrato in Figura 1 in Host Name (or IP address), andremo ad inserire l'indirizzo IP del device raspberry, come porta lasciamo la numero 22 (SSH).

Se i passaggi precedenti sono stati eseguiti correttamente, si aprirà una nuova finestra che chiederà le credenziali per accedere via SSH a raspberry.

Figura 2 Login raspberry

A questo punto se tutti i passaggi sono stati seguiti correttamente siamo collegati con raspberry via SSH.

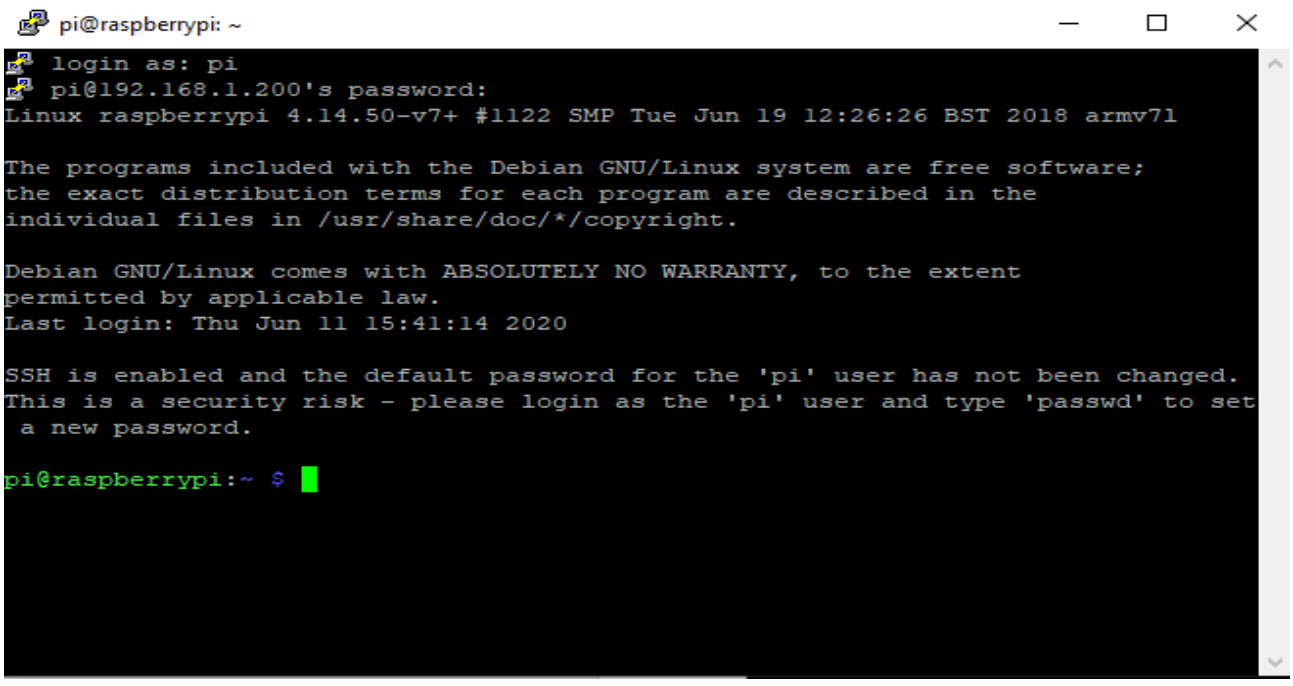

Figura 3 Rasberry root

Per avviare il client per la prima volta bisogna spostarsi nella cartella del progetto e digitare il comando “npm install”, tale comando permette di installare le dipendenze necessarie nella cartella node_modules. A questo punto per avviare il client bisogna digitare il comando “*node main.js”.*

Implementazione Client

Create connection

Prima di ogni cosa viene creato il client opcua tramite la funzione:

```
opcua.OPCUAClient.create
```

A questo punto viene fatto il discovery degli endpoint, per eseguire questa operazione viene chiesto all’utente di inserire l’URL del server. Il programma continuerà a chiedere l’URL fin quando non verrà inserito correttamente.

Dopo aver inserito l’URL del server, verrà effettuato il discovery degli endpoint mediante la funzione:

```
getEndpoints
```

All’utente che usa il client verranno mostrati a video i vari endpoint e le relative informazioni, in particolare verranno mostrati:

- Url dell’endpoint
- Security mode
- Security policy
- Transport protocol
- Security level

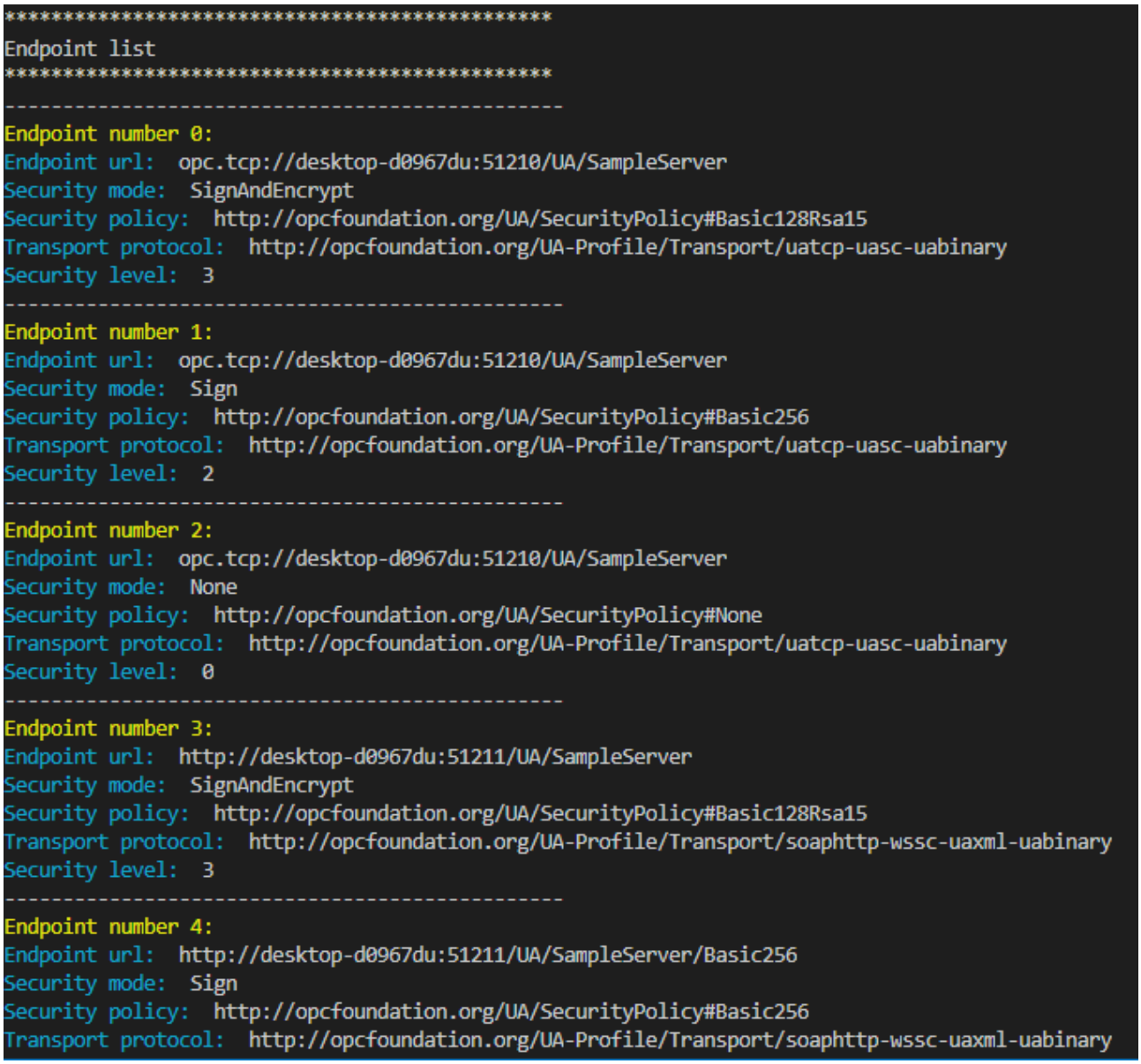

Figura 4 Endpoint list

Si ricordi che il transport protocol supportato è soltanto opc.tcp, pertanto se l'utente cercasse di conettersi ad un endpoint che prevede un transport protocol diverso, verrà stampato un messaggio di errore e verrà chiesto di selezionare un endpoint con il transport protocol supportato (opc.tcp).

A questo punto il client chiederà all'utente di selezionare un solo endpoint dalla lista. Una volta avvenuta la selezione, il client procede alla creazione del canale sicuro in accordo all'attributo Security Mode del Session Endpoint selezionato.

Il codice Javascript inerente alla creazione del canale sicuro è il seguente:

```
new_client = opcua.OPCUAClient.create({
        clientName: "OPCUA JS Client",
        endpoint_must_exist: false,
        securityPolicy:choosen_endpoint.securityPolicyUri,
        securityMode:choosen_endpoint.securityMode,

        connectionStrategy: {

          maxRetry: 1
        }
      })
```

Dal codice si vede che viene istanziato un oggetto OPCUAClient al quale vengono passati come attributi securityPolicy e securityMode rispettivamente il securityPolicy ed il securityMode dell'endpoint selezionato.

Abbiamo tre possibilità:

- Nel caso in cui la security mode dell'endpoint risulti essere **None,** la richiesta per instaurare un canale è inviata senza alcun meccanismo di sicurezza
- Nel caso in cui la security mode dell'endpoint risulti essere **Sign,** la richiesta per instaurare un canale sicuro è inviata usando la chiave privata del certificato dell'applicazione client come firma
- Nel caso in cui la security mode dell'endpoint risulti essere **SignAndEncrypt,** la richiesta per instaurare un canale sicuro è inviata in maniera crittografata utilizzando la chiave pubblica del certificato dell'applicazione server

Dopo aver selezionato un endpoint, la connessione verrà stabilità e l'utente si troverà davanti il menu principale.

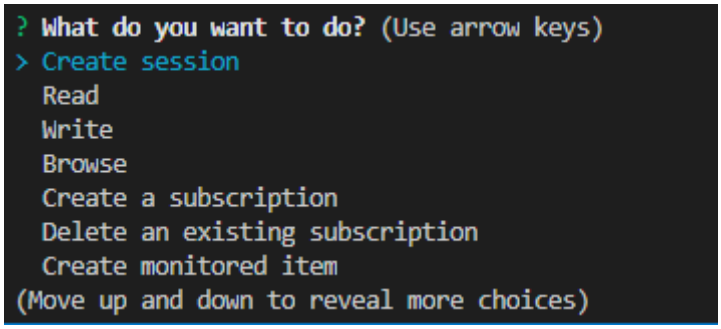

Figura 5 Principal menù

Per realizzare il menù in Figura 5, si è utilizzato Inquirer.

Create session

Una volta instaurato il canale sicuro è possibile procedere con la creazione di una sessione.

Nel client la sessione viene creata con il metodo *createSession()* dell'oggetto OPCUAClient. Tale metodo, infatti, permette di creare e di attivare la sessione ritornando un oggetto appartenete alla classe ClientSession.

```
const session = await client.createSession();
```

Il metodo createSession() vuole come parametro un oggetto di tipo UserIdentityInfo mediante il quale si specificano le credenziali dell'utente che è associato al client. In questo caso, poiché non viene passato nessun parametro, allora l'utente viene considerato di default Anonymous.

Una volta attivata la sessione il client può effettuare delle operazioni attraverso il quale accede ai dati del server.

Read

Attraverso la Read è possibile leggere i valori degli attributi dei nodi indicati dall'utente.

La operazione di lettura è possibile grazie al servizio **Read**[1] degli **Attribute Service Set** [2]

Per effettuare la Read il client richiede all'utente:

- **MaxAge** il quale vincola il server a restituire un valore che abbia un'età (espressa in millisecondi) più piccola di tale parametro
- **Namespace index** che indica in quale namespace dove si trova il nodo
- **Node ID** mediante il quale si identifica il nodo da leggere all'interno dell'namespace index

L'AttributeId è stato impostato di staticamente all'interno del codice come "*Value*".

Di seguito è riportato un esempio di lettura di un nodo avente come NodeClass *Variable*:

```
Insert the max age 100
Insert the namespace Index of the node that you want to read 2
Insert the node ID 10849
--------------------------------------------------
DataType: Int32
Value: 1889072987
Status Code: Good
Source Timestamp: Tue Jun 23 2020 17:02:15 GMT+0200 (GMT+02:00)
Server Timestamp: Tue Jun 23 2020 17:02:15 GMT+0200 (GMT+02:00)
--------------------------------------------------
```

Figura 6 Read

Come si può vedere la Read restituisce i seguenti parametri:

- **DataType**: Tipo di dato che si è andato a leggere
- **Value:** Il valore letto
- **Status Code:** Indica l'esito dell'operazione di read. In questo caso (Status Code == Good) l'operazione di lettura è andata a buon fine ed il valore tornato risulta attendibile.

[1] https://reference.opcfoundation.org/v104/Core/docs/Part4/5.10.2/
[2] https://reference.opcfoundation.org/v104/Core/docs/Part4/5.10.1/

- **Source Timestamp:** timestamp relativo all'istante in cui è stato prodotto il valore
  Server Timestamp: timestamp relativo all'istante in cui il server ha ricevuto il valore

Write

Mediante l'operazione di Write è possibile scrivere un valore per l'attributo Value di un nodo di tipo NodeClass Variable.

La scrittura è implementabile grazie al servizio **Write**[3] degli AttributeServiceSet

I parametri richiesti dal client all'utente per effettuare la write sono i seguenti:

- **Namespace Index** e **NodeID** per identificare il nodo all'interno dell'Address Space
- Il **Valore** che si vuole scrivere

Di seguito vediamo un esempio in cui si prova a scrivere un nodo di tipo NodeClass Variable e avente come attributo Value una stringa.

```
Insert the max age 100
Insert the namespace Index of the node that you want to read 2
Insert the node ID 10227
-------------------------------------------------
DataType: String
Value: 빨간] 검정. 뱀 코끼리 원숭이 석회 석회
Status Code: Good
Source Timestamp: Tue Jun 23 2020 17:01:00 GMT+0200 (GMT+02:00)
Server Timestamp: Tue Jun 23 2020 17:49:26 GMT+0200 (GMT+02:00)
-------------------------------------------------
```

Figura 7 Write

[3] https://reference.opcfoundation.org/v104/Core/docs/Part4/5.10.4/

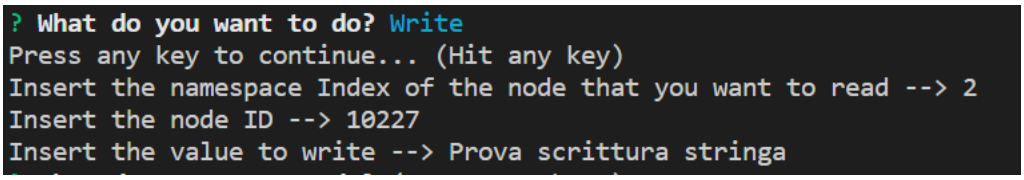

Figura 8 Write check

Nell'esempio si vuole andare a scrivere sul nodo il valore "Prova scrittura stringa"

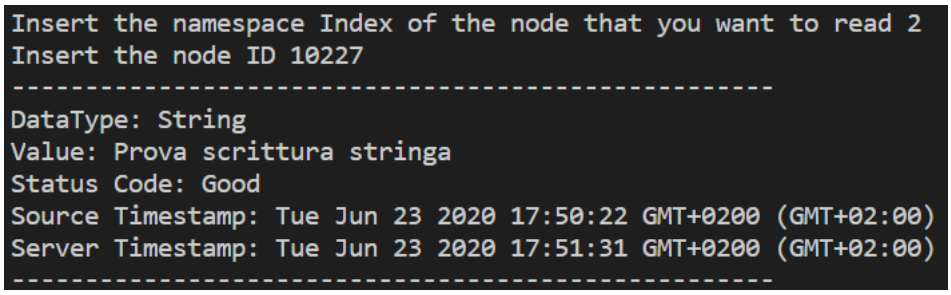

Figura 9 Write a string

Dopo aver effettuato la write, andando a leggere nuovamente il nodo, si nota che il valore è stato cambiato correttamente.

Browsing

Come prerequisito, è necessario che sia già stata creata una sessione con il server, nel nostro caso specifico, qualora non fosse stata ancora creata una sessione, il client mostrerà il seguente messaggio di errore:

"*Maybe you don't have created any session, please create a session!*"

Il servizio di browsing è utilizzato dal client per visitare l'address space del server.

```
var folders = [];
var root = "RootFolder"
var path = []

try {
  var browseResult = await session.browse(root);
} catch{
  console.error("Maybe you don't have created any session, please create a session!");
  return "back";
}
```

Figura 10 Browse root folder

Come mostrato in Figura 10, richiamiamo il metodo browse sulla session precedentemente creata e passiamo come parametro il nodo di cui si vogliono avere le ***references.*** In figura 10, si evidenzia il fatto che il nostro nodo di partenza per iniziare l'operazione di browsing sia ***RootFolder***. I nodi collegati a ***RootFolder***, sono:

- Object
- Types
- Views

Infatti, se venisse selezionata l'opzione ***browse*** dal menù principale, la schermata che si presenterebbe all'utente che sta utilizzando il client, sarebbe la seguente:

```
? What do you want to do? Browse
? Select the directory in which you want to navigate (Use arrow keys)
> Objects
  Types
  Views
  Insert the Node ID manually
  Return to principal menù
```

Figura 11 Browse first level

A questo punto, attraverso il menù implementato grazie ad Inquirer, possiamo selezionare una delle cinque opzioni, di seguito alcuni screenshots dimostrativi:

Browse Objects:

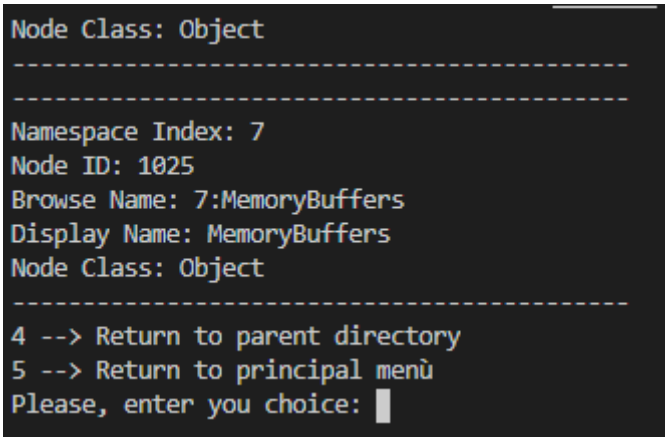

Figura 12 Browse Objects

A questo punto, è possibile decidere se si vuole tornare al menù principale o se si vuole ritornare al livello precedente e continuare eventualmente l'operazione di browsing.

Browse Types

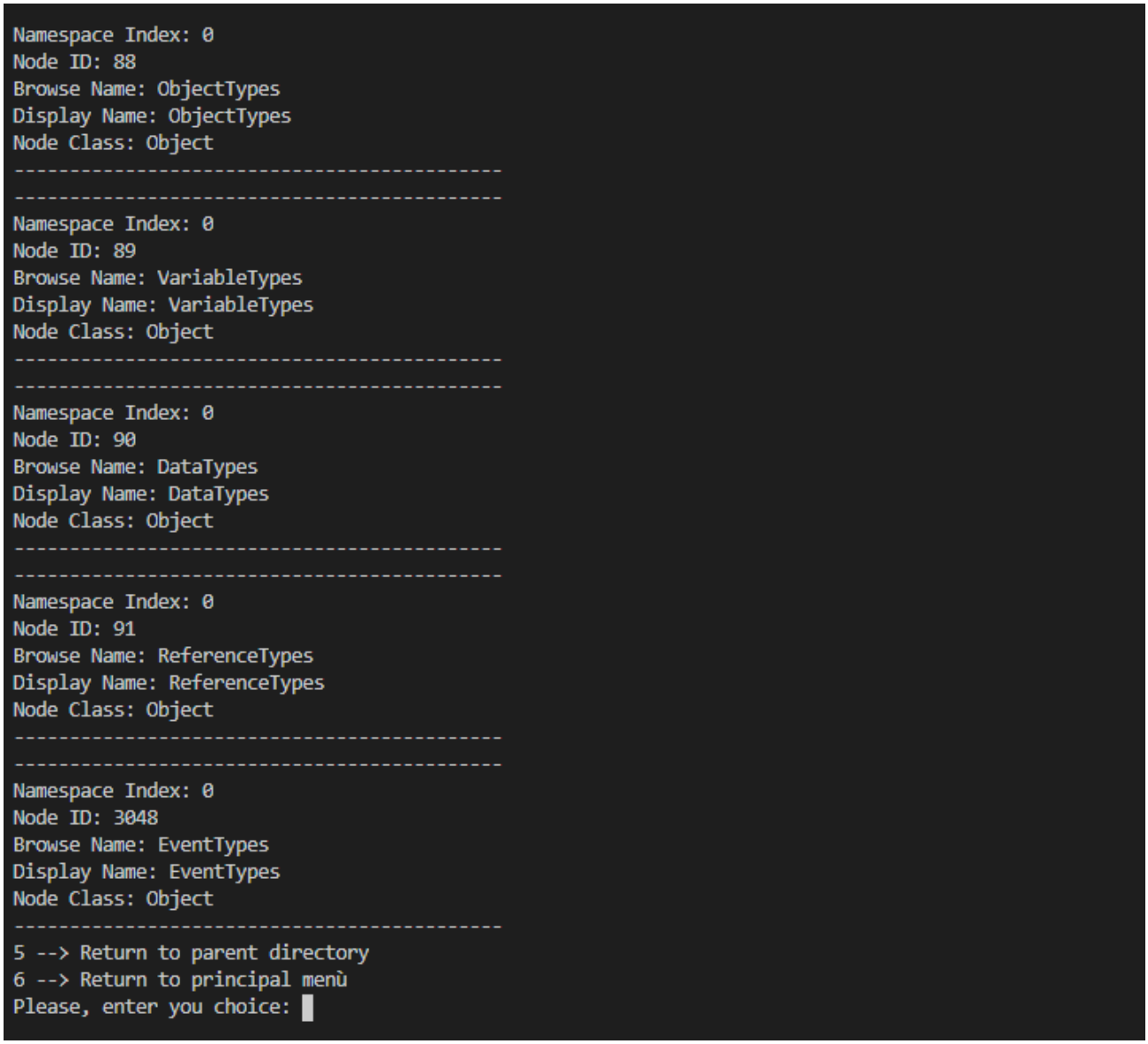

Figura 13 Browse types

In alternativa si può scegliere di effettuare il browsing inserendo i valori di **Namespace Index** e **Node ID** manualmente.

Ad esempio supponiamo di voler fare il browsing di **Data:**

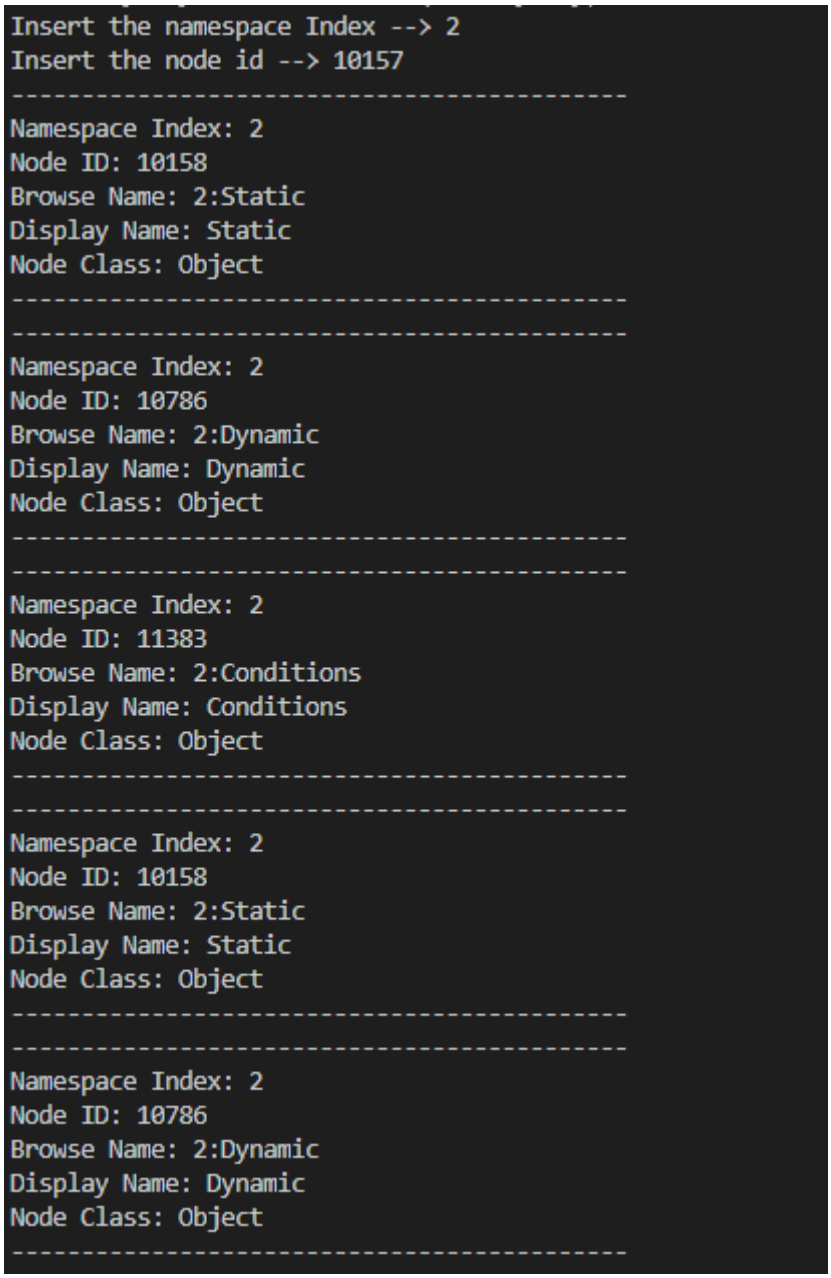

Figura 14 Browse Data

Create subscription and MonitoredItem

Oltre ai servizi read/write, esiste un'alternativa più potente ed elegante per poter accedere ai dati. Per ogni session attiva, il client può creare una o più subscription. Tale servizio permette al client OPC di effettuare la sottoscrizione a nodi di suo interesse, in particolare il client nell'ambito di una subscription può creare uno o più monitored item.

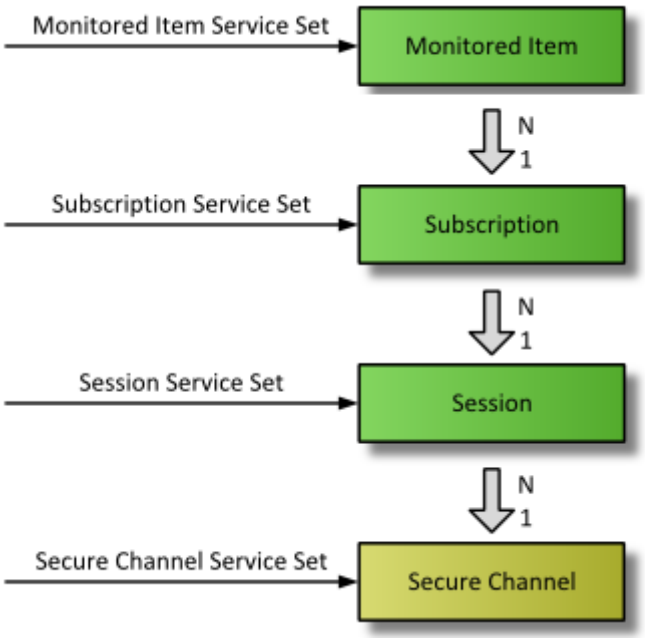

Figura 15 Subscription scheme

In generale in una sessione possono essere attive più subscription, per ogni subscription si possono creare più monitored item. Sarà compito del server inviare le notifications al client che ha effettuato la subscription per quel particolare nodo.

I tipi di cambianti per i quali il client può ricevere i notification message sono i seguenti:

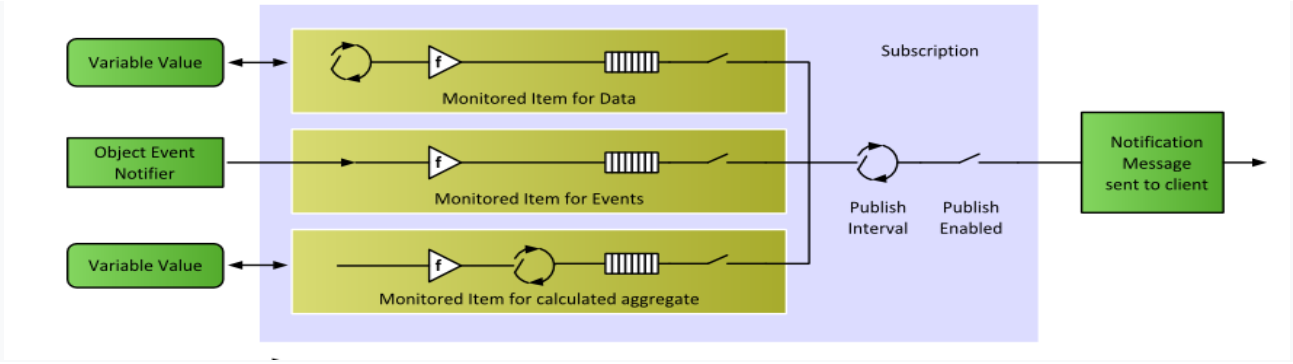

Figura 16 Types of changes

Variable value: viene monitorato l'attributo value di una variabile, se si verifica un cambiamento del valore, del timestamp o dello status code, il monitored item invierà una notification.

Attributi: un qualsiasi attributo di un qualunque nodo può essere associato ad un monitored item. Quando il valore dell'attributo cambia, il monitored item invierà una notification.

Object Event Notifier: i nodi appartenenti al NodeClass Object possono essere associati ad un monitored item, in questo caso si andrà a monitorare il verificarsi di un determinato evento.

Di seguito il funzionamento del servizio implementato nel nostro client:

Il requisito fondamentale per creare una subscription è che sia stata già creata una sessione a cui associare la subscription, se non è stata creata alcuna sessione, il client informerà l'utente e lo inviterà a crearla.

Per creare una nuova subscription, è necessario selezionare la voce "*create a subscription"* dal menù principale, fatto ciò il client chiederà all'utente di inserire i parametri della subscription, in particolare:

Requested publishing interval: serve ad impostare l'intervallo di pubblicazione delle notifiche desiderato dal client, qualora non venisse inserito, il valore di default sarebbe 1 secondo

RequestedMaxKeepAliveCount: definisce il numero di publishing interval che possono trascorrere senza che il client riceva alcuna notifica, trascorso questo tempo il server manderà un keepAlive message per dire che la subscription è attiva nonostante non siano state inviate notifications. Il valore di default è 5.

RequestedLifeTimeCount: definisce il numero di publishing interval che possono trascorrere senza che venga monitorata alcuna attività del client. Scaduto questo tempo la subscription viene eliminata e vengono liberate le relative risorse. Valore di default 40.

MaxNotificationsPerPublish: indica al server il numero massimo di notification da inserire in un notification message. Valore di default 10.

Priority: definisce la priorità della subscription rispetto alle subscription già create, tale priorità serve al server per decidere quale notification message inviare prima.

Dopo aver creato la subscription, è possibile sempre dal menù principale creare un monitored item. Per creare un monitored item deve essere stata creata almeno una subscription. Dopo aver selezionato "*create monitored item*", il client farà scegliere tramite un menù la subscription, mediante il subscription ID, a cui associare il monitored item.

A questo punto verrà chiesto all'utente di inserire:

- **Namespace Index**
- **NodeID**
- **Sampling interval:** definisce la frequenza con cui il server campiona il monitored item
- **Queue size:** dimensione della coda

Di seguito uno screenshot dimostrativo:

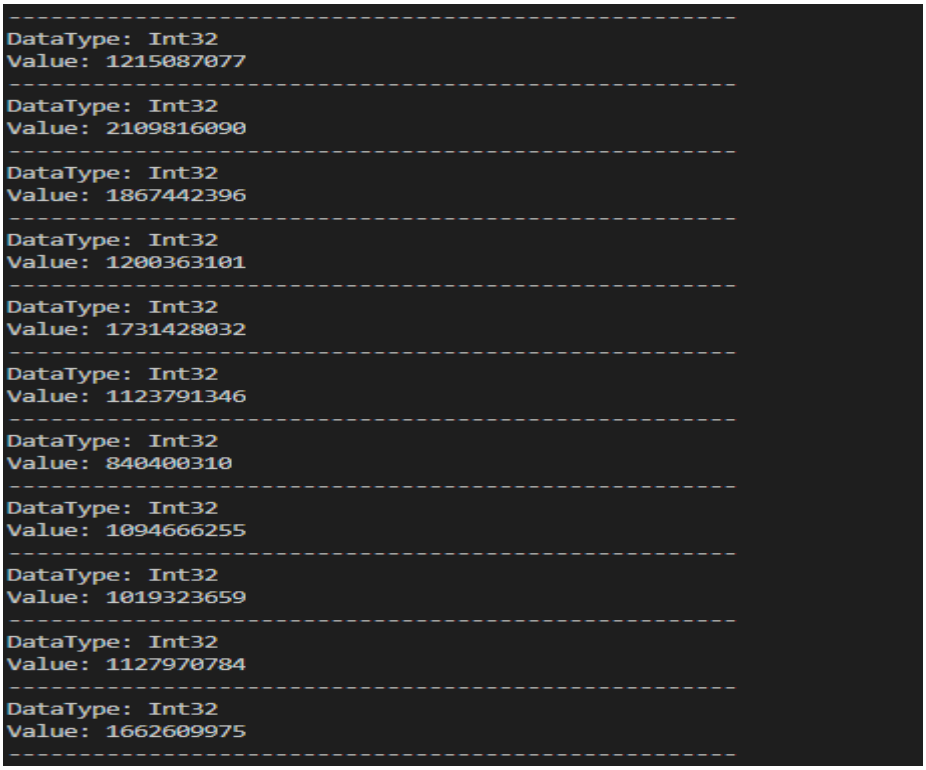

Figura 17 Monitored Item

In questo esempio (figura 17) è stato monitorato il nodo caratterizzato da Namespace Index: 2 e Node ID: 10849, tale nodo corrisponde ad un INT 32 dinamico, ogni volta che tale nodo cambia valore viene inviata una notification ed il client stamperà a video il valore. Per terminare il monitoraggio basta cliccare CTRL+C e il client tornerà al menù principale.

Delete an exisisting subscription

Selezionando questa voce dal menù principale, il client consentirà all'utente di scegliere da una lista quali delle subscription esistenti eliminare. Se non è presente alcuna subscription il client stamperà a video il messaggio: *"No subscription founded".*