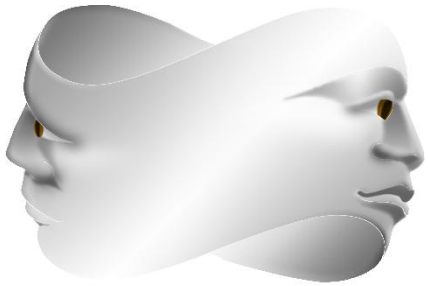



# OPC-UAC 2.0

## MANUALE OPERATIVO

Libreria OPC UA Client per applicazioni Windows.

Dichiarazioni di marchio registrato

Automa, OPC-UAC sono marchi registrati di Automa srl

Tutti gli altri marchi registrati non esplicitamente dichiarati, sono di proprietà delle rispettive società.

# INDICE

# 1 Introduzione

OPC-UAC è una libreria per la connessione di applicazioni Windows con i server OPC UA.

OPC-UAC è a disposizione nella formula commerciale RUNTIME; la licenza è singola per una sola applicazione.
Il funzionamento della libreria è legato alla presenza di una specifica attivazione di tipo hardware o software; in assenza di tale attivazione la comunicazione sarà limitata a soli 15 minuti.
Per la gestione dell'attivazione fare riferimento al paragrafo "Attivazione licenza della libreria" a pag. 5.

Dopo l'installazione del prodotto, per un primissimo test fare riferimento alle applicazioni demo descritte al paragrafo "Applicativi demo" a pag. 45.

# 2 Caratteristiche

La libreria fornisce un'interfaccia semplice che supporta operazioni di lettura, scrittura e monitor delle variazioni dei valori delle variabili.

La libreria OPC-UAC supporta:
- connessione con un numero illimitato di server OPC UA
- discovery server su host
- modalità di connessione:
    - Trasporto: UA-TCP UA-SC UA-Binary
    - Security policy: None, Basic256, Aes128-Sha256-RsaOaep, Basic256Sha256, Aes256-Sha256-RsaPss
    - Autenticazione utente: Anonymous, UserName
    - Gestione certificati self-signed
- Servizio "Data Access"
    - lettura e scrittura in modalità singola e array monodimensionale dei seguenti tipi di valori:
        - Boolean
        - Byte e SByte
        - UInt16 e Int16
        - UInt32 e Int32
        - Single (Float)
        - Double
        - String
        - ByteString
        - LocalizedText (solo 'Default language')
    - lettura delle informazioni accessorie:
        - qualità del dato
        - server timestamp
        - source timestamp
    - gestione della scrittura in modalità "SafeWrite"
    - scrittura array con modalità "IndexRange"
    - monitor della variazione dei valori (DataChange Subscriber)
- Servizio "Esecuzione Metodi" (solo tramite interfaccia .NET)

# 3 Requisiti minimi

I requisiti minimi per il funzionamento della libreria OPC-UAC sono quelli richiesti dal sistema .NET Framework 4.6.2.
In base alle informazioni fornite da Microsoft la versione minima del Sistema Operativo è:

- Client - Microsoft Windows 7 SP1
- Server - Windows Server 2008 R2 SP1

# 4 Limiti

Rispetto le funzionalità di base si segnalano per OPC-UAC i seguenti limiti:

- Non supporta i seguenti tipi di dati:
  - Int64
  - UInt64
  - DateTime
  - Guid
  - XmlElement
  - NodeId
  - ExpandedNodeId
  - StatusCode
  - QualifiedName
  - ExtensionObject
  - DataValue
  - Variant
  - DiagnosticInfo
  - Number
  - Integer
  - UInteger
  - Enumeration
- Valori array: sono supportati solo array monodimensione

# 5 Competenze

OPC-UAC richiede, per l'utilizzo, le seguenti competenze:

- conoscenza minima della tecnologia e terminologia OPC UA
- conoscenza minima della gestione dei certificati
- conoscenza minima delle comunicazioni Ethernet TCP/IP

# 6 Note di rilascio rispetto alla versione 1.0 (Marzo 2021)

## 6.1 Versione 2.0 (Luglio 2025)

Nuove funzionalità:

- Supporto autenticazione utente con Username e Password
- Facilitazione per collegamento con server senza sicurezza
- Supporto del tipo LocalizedText (solo 'default language')
- Supporto chiamata ai metodi
- Supporto scrittura array con modalità IndexRange (modifica parziale)

Problemi risolti:

- Tool configuratore: possibile blocco durante il browse delle risorse del server
- Runtime: possibile ritorno temporaneo di un vecchio valore dopo ripetute scritture

# 7 Installazione

OPC-UAC viene distribuito su CD-ROM ed è anche prelevabile dal sito http://www.automa.it nella sezione "FreeDownload".

Per attivare la procedura di installazione:

1. eseguire il programma "Setup_OPC-UAC_Runtime.exe". Il programma è presente nel CD-ROM o nei file scaricati da internet.

2. l'installazione avviene in modo quasi completamente automatico. All'utente viene richiesta la cartella in cui installare OPC-UAC, per default viene proposta la cartella "\Programmi (x86)\Automa\Communication Tools\OPC-UAC 2.0".
   Gli applicativi dimostrativi sono invece installati nella cartella "<APPDATA>\Automa\Communications Tools\OPC-UAC 2.0" per evitare problemi legati alla gestione dello UAC. Il path in cui si trova la cartella "<APPDATA>" dipende dal Sistema Operativo che si sta utilizzando; per accedervi velocemente è possibile aprire una finestra "Esplora file" e inserire nella barra degli indirizzi il nome simbolico "%APPDATA%"..

Nella cartella di installazione verranno create tre sottocartelle:

| Cartella | Contenuto |
|---|---|
| DRIVER | file della libreria OPC-UAC e tool configuratore |
| USER GUIDE | questa documentazione in formato PDF |
| TOOLS | Tools per la gestione delle chiavi hardware e software e file di redistribuzione di componenti Microsoft. |

Nella cartella "<APPDATA>\Automa\Communication Tools\OPC-UAC 2.0" verrà creata la sottocartella:

| Cartella | Contenuto |
|---|---|
| DEMO | Applicativi dimostrativi e programma configuratore. |

Nella cartella di sistema "Program Data" (default: "C:\ProgramData") viene inoltre creata la cartella "Automa" dove risiede il "Certificate Store"; qui verranno generati gli "Application Certificate" e verranno salvati eventuali certificati dei server OPC UA utilizzati.
Lo store è costituito da un insieme di cartelle con la struttura tipica di un sistema PKI (Public Key Infrastructure):

- cartella Own : sono presenti i certificati relativi al tool configuratore e al driver runtime
- cartella Rejected: sono presenti eventuali certificati dei server che non sono stati accettati
- cartelle Trusted: sono presenti i certificati dei server che sono stati validati

L'installatore, se necessario, installerà i componenti ".NET Framework 4.6.2 Runtime" e "Microsoft Visual C++ 2015-2022 Redistributable (x86)".

Nel menu "Start" di Windows viene generato il gruppo "Programmi" ▶ "Automa" ▶ "Communication Tools" ▶ "OPC-UAC 2.0" e sul desktop viene creato l'elemento denominato "Automa - Super-Flash OPC-UAC 2.0" contenente i link per l'accesso rapido.

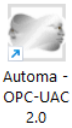
# 8 Utilizzo della libreria

Di seguito le informazioni necessarie per l'utilizzo della libreria.

## 8.1 Deployment

Nella cartella "DRIVER" di installazione sono presenti i file necessari all'esecuzione del tool configuratore e delle applicazioni.

Dall'elemento creato sul desktop:




è possibile accedere alla cartella selezionando il link "Driver".

Nella cartella sono presenti i seguenti file:

| Ambito | File | Descrizione |
|---|---|---|
| Tool configuratore | OPCUACCFG.EXE | Tool configuratore di OPC-UAC |
| | Opc.Ua.ClientControls.dll | |
| | Opc.Ua.QuickstartsLibrary.dll | |
| | OPCUACCFG.Config.XML | File contenente parametri che regolano il funzionamento del tool configuratore. |
| Libreria | OPCUAC.DLL | Modulo principale della libreria OPC-UAC |
| | OPCUAC.Config.XML | File contenente parametri che regolano il funzionamento della libreria. |
| Comuni | AOPCUALibrary.DLL | File DLL comuni alla libreria e al tool configuratore. |
| | BouncyCastle.Crypto.DLL | |
| | Opc.Ua.Client.ComplexTypes.DLL | |
| | Opc.Ua.Client.DLL | |
| | Opc.Ua.Configuration.DLL | |
| | Opc.Ua.Core.DLL | |

Per l'esecuzione del tool di configurazione è necessario:

- copiare in una cartella i file di **Tool configuratore** e **Comuni**

Per l'esecuzione dell'applicazione è necessario:

- copiare i file di **Libreria** e **Comuni** nella stessa directory in cui risiede il file eseguibile
- copiare il file di configurazione della basedati OPCUAC.XML (predisposto tramite il tool configuratore) nella directory di lavoro dell'applicazione

Attenzione: se sulla macchina target finale non viene installato il prodotto occorre assicurarsi che nel sistema siano presenti i seguenti componenti Microsoft:

- Microsoft .NET Framework 4.6.2
- Microsoft Visual C++ 2015-2022 Runtime Files

Nella cartella "TOOLS" di installazione sono disponibili due eseguibili forniti da Microsoft per l'installazione manuale dei due componenti, rispettivamente:

- NDP462-KB3151800-x86-x64-AllOS-ENU.exe
- VC_redist.x86.exe

Il "Certificate Store", se non predisposto preventivamente, viene creato automaticamente alla prima esecuzione del tool di configurazione o dell'applicazione.

## 8.2 Attivazione licenza della libreria

Il funzionamento continuativo della libreria OPC-UAC è legato alla presenza di una specifica attivazione di tipo hardware o software. In assenza di tale attivazione il funzionamento è limitato a sessioni di 15 minuti.

Sono supportate due tipologie di chiavi, ognuna delle quali richiede una specifica procedura di installazione:

**Chiave hardware**: dispositivo hardware USB che non richiede l'installazione di alcun software di gestione.

**Chiave software**: file "OPCUAC.CTK" contenente il codice di attivazione.
La chiave software è strettamente legata al singolo PC e deve essere obbligatoriamente attivata sul PC in cui verrà eseguita l'applicazione (macchina target).
Nella fase di sviluppo, se si utilizza un PC diverso, OPC-UAC può essere utilizzato senza chiave in modalità dimostrativa (sessioni di 15 minuti); altrimenti (ad esempio nel caso sia necessario effettuare un test dell'applicazione che richieda il funzionamento continuativo per un tempo maggiore di quello della singola sessione dimostrativa di 15 minuti) è necessario dotarsi di una seconda chiave software.
La chiave di protezione software richiede la presenza di una scheda di rete Ethernet.

Per l'utilizzo della chiave software è necessario seguire la seguente procedura:

1. sul PC in cui si intende attivare il driver eseguire il tool "WRFCCode.exe" e rilevare il codice cliente (Customer Code)
2. comunicare il codice cliente (Customer Code) via e-mail ad Automa (sales@automa.it); la quale provvederà a generare il codice di attivazione e fornire il file "OPCUAC.CTK"
3. copiare il file "OPCUAC.CTK" nella directory principale dell'applicazione

Il tool "WRFCCode.exe" è presente nella cartella "TOOLS" del path di installazione del driver ed è disponibile nella sezione free download del sito Automa.

## 8.3 Interfacce della libreria

La libreria è composta dal modulo principale OPCUAC.DLL e una serie di altre DLL dipendenti basate sul .NET Framework v.4.6.2.

La libreria OPC-UAC fornisce due interfacce:

1. interfaccia .NET assembly (namespace OPCUAC): interfaccia preferenziale completa
2. interfaccia "C" (set di funzioni Win32 C): interfaccia con alcune limitazioni

Per l'utilizzo dell'interfaccia :NET in versioni precedenti di Visual Studio e l'utilizzo dell'interfaccia "C" sono forniti anche i seguenti file accessori presenti nei progetti demo OPCUACDemo_VC2010:

- OPCUACINTERFACE.CS: contiene la definizione di costanti, strutture e metodi di supporto all'utilizzo della DLL in applicazioni C# Windows Form tramite P/Invoke.
- OPCUAC.H: contiene costanti, strutture e prototipi delle funzioni
- OPCUAC.LIB: file Import Library (per il link implicito)

Per semplicità verranno utilizzati i termini tipici della programmazione C anche nel caso di riferimenti ad elementi corrispondenti in altri linguaggi; esempio:

- Funzione -> funzione/metodo/delegate
- Struttura -> struttura, classe
- Costante -> valore costante definita tramite la direttiva #define o tramite campo statico di una classe

Nei paragrafi che seguono sono illustrati il principio di funzionamento della libreria, la configurazione della basedati e una descrizione delle funzioni per categoria.

## 8.4 Principio di funzionamento

### 8.4.1 Generale

La libreria OPC-UAC prevede di operare con una basedati preconfigurata contenuta nel file "OPCUAC.XML".
La configurazione della basedati si effettua con il tool configuratore in cui si definiscono le connessioni con i server, le subscription con l'elenco degli item di tipo "Variable" a cui si vuole accedere ed eventuali metodi da eseguire.
Ad ogni item viene associato un identificativo alfanumerico univoco che maschera e virtualizza i riferimenti ai NodeId (identificativo univoco dell'item nel server).
La libreria fornisce un'interfaccia molto semplice che consente di leggere, scrivere e ricevere le notifiche di variazione dei valori degli item di tipo "Variable" ed eseguire la chiamata a metodi.
Le connessioni ai server sono gestite in modo automatico e trasparente, comprese le eventuali riconnessioni.
La gestione dei certificati è fornita con una modalità semplificata: in fase di inizializzazione la libreria genera, se necessario, un certificato self-signed e in fase di connessione accetta automaticamente i certificati self-signed dei server.

Per la gestione dell'attivazione fare riferimento al paragrafo "Attivazione licenza della libreria" a pag. 5.

**Note**
**Attenzione!** La connessione ai server OPC UA è un'operazione che può richiedere alcuni secondi. Le richieste effettuate in questa fase iniziale di connessione possono fallire con uno specifico errore di connessione non stabilita.
Per la descrizione dei possibili codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51.

### 8.4.2 Lettura valori

La libreria mantiene una cache con l'ultimo valore disponibile di ogni item.
Dopo aver stabilito la connessione con il server la libreria esegue per ogni item una lettura per inizializzarne il valore e attiva il servizio di subscription che, al variare del valore, invierà una notifica di aggiornamento al client.

La frequenza di aggiornamento di un item è determinato dal "Sampling Interval" del singolo item e dal parametro "Publishing Interval" configurato nella subscription che contiene l'item.

Se il server non supporta il servizio di subscription ad ogni richiesta la libreria effettuerà una lettura del dato dal server.
La libreria consente anche di forzare la rilettura del valore dal server.

### 8.4.3 Scrittura valori

L'operazione di scrittura viene effettuata in modo bloccante per il tempo necessario a garantire la consegna del valore alla periferica o determinarne l'eventuale impossibilità.
Solo in casi particolari, quando è anche attiva la modalità "Safe Write", la procedura può impegnare del tempo aggiuntivo. Vedere le note al paragrafo "Modifica di un item" a pag. 16.
Nel caso di item di tipo array la libreria utilizza, in base agli elementi da aggiornare e alle capacità del server, le seguenti modalità:

- scrittura immediata dell'intero valore: si effettua quando è richiesta la modifica di un numero di elementi uguale o maggiore alla dimensione attuale dell'array.
- scrittura parziale con la modalità "IndexRange": si effettua quando è richiesta la modifica di solo alcuni elementi e il server supporta tale modalità.

- scrittura con la modalità RMW: si effettua quando è richiesta la modifica di solo alcuni elementi e il server non supporta la modalità "IndexRange". La procedura prevede una lettura preventiva dell'intero valore, la modifica in memoria degli elementi e la riscrittura dell'intero valore.

**Note**

**Attenzione!** Dalla versione 2.0 il driver utilizza di default la modalità "IndexRange" (se disponibile sul server).
Se necessario è possibile disabilitare questa funzionalità editando direttamente, con uno standard editor di testo, il file di configurazione "OPCUAC.XML" aggiungendo o modificando per il server interessato il parametro "WriteFullArrayOnly" con valore "true".
Per individuare il punto di modifica occorre individuare la riga con il parametro "UserDescription" contenente la descrizione del server interessato (nell'esempio "Plant PLC") e aggiungere o modificare il parametro come indicato di seguito.

```
<COPCCServerCfg>
  <DisableDomainCheck>false</DisableDomainCheck>
  <SafeWriteTimeout>1000</SafeWriteTimeout>
  <SessionTimeout>20000</SessionTimeout>
  <UseSecurity>true</UseSecurity>
  <UserDescription>Plant PLC</UserDescription>
  <WriteFullArrayOnly>true</WriteFullArrayOnly>
```

Nel tool configuratore, nel riquadro in alto a destra, in corrispondenza dell'attributo "Array" è indicato se il dato supporta o meno la scrittura parziale.

Esempio:


### 8.4.4 Chiamata a metodi

La libreria fornisce un'interfaccia per eseguire la chiamata ai metodi gestendone gli argomenti di input/output e l'esito.

## 8.5 Configurazione della basedati

La libreria OPC-UAC basa il proprio funzionamento sulle informazioni contenute nel file di configurazione "OPCUAC.XML".
L'attività di configurazione si effettua tramite il programma di configurazione "OPCUACCFG.EXE" e deve essere svolta principalmente in una condizione di on-line con i server OPC UA che si intende utilizzare.

La configurazione è composta dai seguenti elementi:

| | |
|---|---|
| **Server:** | rappresenta la connessione ad un OPC UA server |
| **Subscription:** | è un contenitore di item di tipo (node class) "Variable" associato al server. Il server può contenere più subscription. |
| **Methods:** | è un contenitore predefinito di item di tipo (node class) "Method" associato al server. |
| **Item:** | identifica una risorsa di tipo (node class) "Variable" presente in un server OPC UA di cui è possibile leggere e scrivere il valore. Gli item sono contenuti in una subscription. |
| **Metodo:** | identifica una risorsa di tipo (node class) "Method" presente in un server OPC UA che rappresenta una funzione che può essere eseguita su un'oggetto. I metodi sono contenuti nel gruppo predefinito "Methods". |

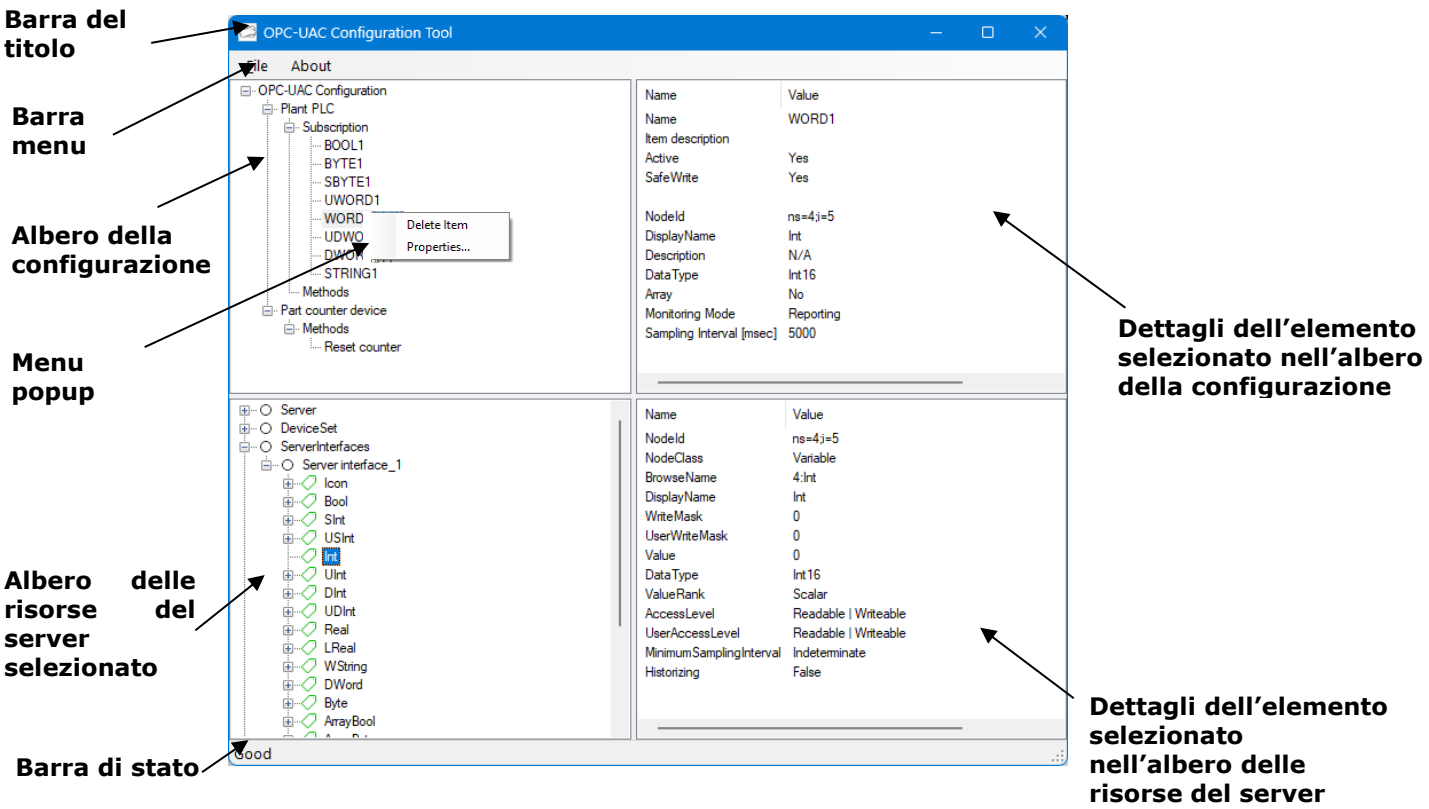
### 8.5.1 Programma di configurazione

Per l'esecuzione del programma di configurazione "OPCUACCFG.EXE" fare riferimento alle indicazioni presenti al paragrafo "Deployment" a pag. 4.

Nella fase di inizializzazione il configuratore carica la configurazione corrente contenuta nel file "OPCUAC.XML"; nel caso di file mancante o errore nella lettura il configuratore crea una configurazione vuota.

All'avvio del configuratore viene visualizzata la finestra principale composta da:

- Barra del titolo
- Barra del menu
- Barra di stato
- Area albero della configurazione: in questa area viene visualizzato l'albero dei server/subscription/item/metodi che compongono la configurazione
- Area dettagli elemento di configurazione: in questa area sono visualizzate le informazioni di dettaglio relative all'elemento di configurazione selezionato
- Area albero delle risorse del server
- Area dettagli risorsa del server
- Popup menu




Per tutto il tempo di esecuzione il programma configuratore cerca di stabilire e mantenere la connessione con tutti i server presenti nella configurazione in modo da consentire il browsing degli item disponibili nella basedati dei server e la loro aggiunta nella basedati del driver.

Selezionando nell'albero della configurazione un qualsiasi elemento di un server viene visualizzata nella barra di stato l'informazione sullo stato della connessione.
Se la connessione è attiva (stato "Good"), nei riquadri inferiori sono visualizzati rispettivamente a sinistra l'albero degli item disponibili nel server e a destra un dettaglio di alcuni attributi dell'eventuale item selezionato.

Se la connessione non è attiva (stato "Server not connected") i riquadri rimangono vuoti; fare riferimento al paragrafo "Troubleshooting" a pag. 53.

Di seguito sono elencati tutti i comandi disponibili con le relative modalità di accesso.
Nei paragrafi successivi sono descritte le funzionalità dei singoli comandi.

| Comando | Modalità attivazione | Tasto |
|---|---|---|
| Salva | Barra menu: "File" ▶ "Save" | CTRL+S |
| Esci | Barra menu: "File" ▶ "Exit" | ALT+F4 |
| Informazioni | Barra menu: "About" ▶ "Informazioni" | CTRL+X |

| Comando | Modalità attivazione | Tasto |
|---|---|---|
| Creazione server | Menu popup elemento radice ("OPC UAC Configuration"): "New server…" | INS (*) |
| Creazione subscription | Menu popup elemento server: "New subscription…" | INS (*) |
| Creazione item | Operazione Drag&Drop del singolo elemento di tipo "Variable" dall'albero delle risorse all'albero della configurazione (in una subscription dello stesso server). | |
| Creazione metodo | Operazione Drag&Drop del singolo elemento di tipo "Method" dall'albero delle risorse all'albero della configurazione (nel gruppo "Methods" dello stesso server). | |
| Cancella server / subscription /item/metodo | Menu popup elemento selezionato: "Delete…" | DEL (*) |
| Proprietà server / subscription / item/metodo | Menu popup elemento selezionato: "Properties…" | CTRL+P (*) |

**Note**
**(*)** Comandi contestuali rispetto all'elemento selezionato.

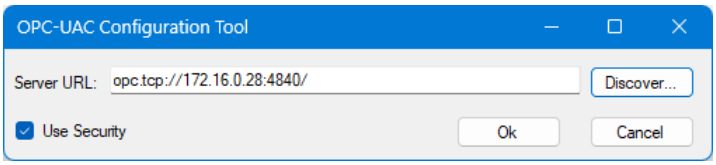
Partendo da una configurazione vuota gli step minimi necessari sono:
1. creare un server
2. creare una subscription nel server
3. creare gli item nella subscription

### 8.5.2 Configurazione di un server

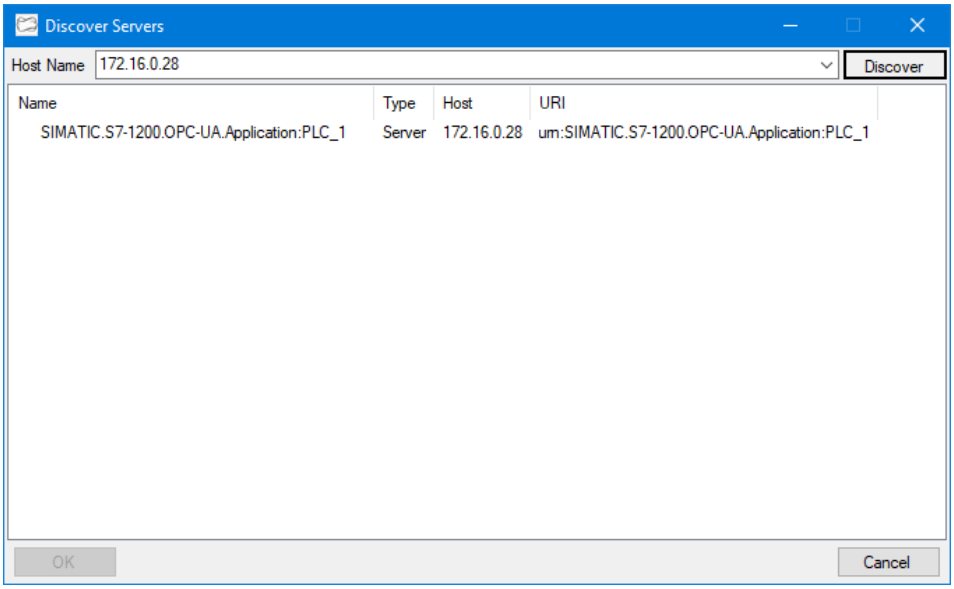
Dall'albero della configurazione è possibile accedere ai comandi per creare un nuovo server oppure modificare o cancellare un server esistente.

#### 8.5.2.1 Creazione di un server

Per la creazione di un nuovo server viene visualizzata la seguente finestra.




È possibile inserire direttamente nel campo "Server URL" la stringa URL del server da contattare oppure è possibile accedere all'opzione "Discover…" che consente di effettuare la ricerca dei server OPC UA disponibili su un determinato host.




Nella finestra "Discover Servers" inserire il nome o l'indirizzo IP dell'host che ospita il server che si intende contattare e selezionare il comando "Discover".
Se la ricerca va a buon fine nell'elenco si visualizzano i server disponibili; selezionando il server interessato e proseguendo con "OK" si ritorna alla finestra principale dove viene inserita automaticamente la stringa URL del server selezionato.

Nella finestra principale è presente la flag "User Security" che consente di indicare la preferenza sull'uso o meno della sicurezza: confermando con "OK" viene verificata la correttezza sintattica dell'URL inserito e nel caso di esito positivo si innesca un tentativo di connessione al server.

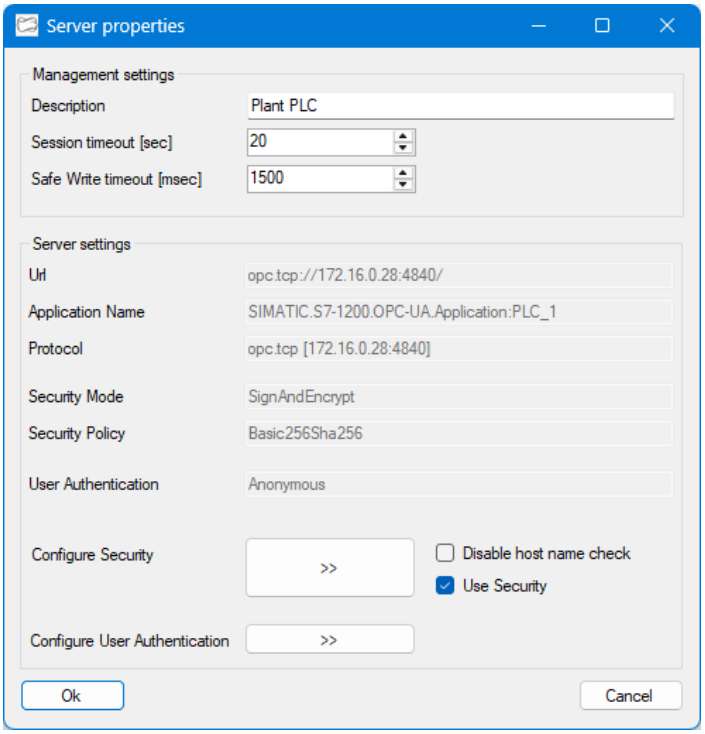
L'operazione può richiedere alcuni secondi; il cursore del mouse assume la forma di 'wait' e nella status bar viene visualizzato un messaggio informativo.

Se il tentativo di connessione fallisce viene visualizzato un messaggio di errore e l'operazione di creazione del server viene annullata, altrimenti la procedura continua con la visualizzazione della finestra delle proprietà in cui è possibile inserire altri parametri e modificare le impostazioni assegnate di default durante la connessione iniziale.
Per la descrizione della finestra delle proprietà fare riferimento al paragrafo seguente.

Se al termine della configurazione risulta già presente un server con le stesse caratteristiche, verrà segnalato un errore e il server non verrà aggiunto alla configurazione.

### 8.5.2.2 Modifica di un server

Dopo il comando di creazione o modifica di un server si visualizza la finestra delle proprietà:




Le proprietà sono suddivise in due gruppi:

- "Management settings": parametri che regolano alcuni aspetti di gestione del lato client
- "Server settings": parametri per la configurazione della connessione con il server (protocollo, impostazioni di sicurezza dei livelli di trasporto, applicazione ed utente)

Parametri "**Management settings**".

| Description |
|---|
| Descrizione libera da associare al server. |
| **Session timeout** |
| Timeout massimo di attesa per la creazione di una sessione di comunicazione con il server.<br><br>La creazione di una sessione comporta l'esecuzione di varie operazioni e lo scambio di numerose informazioni tra client e server; tipicamente il tempo necessario è nell'ordine di alcuni secondi. |
| **Safe Write timeout** |
| Timeout massimo di attesa per la rilettura del dato aggiornato a seguito di un'operazione di scrittura. |

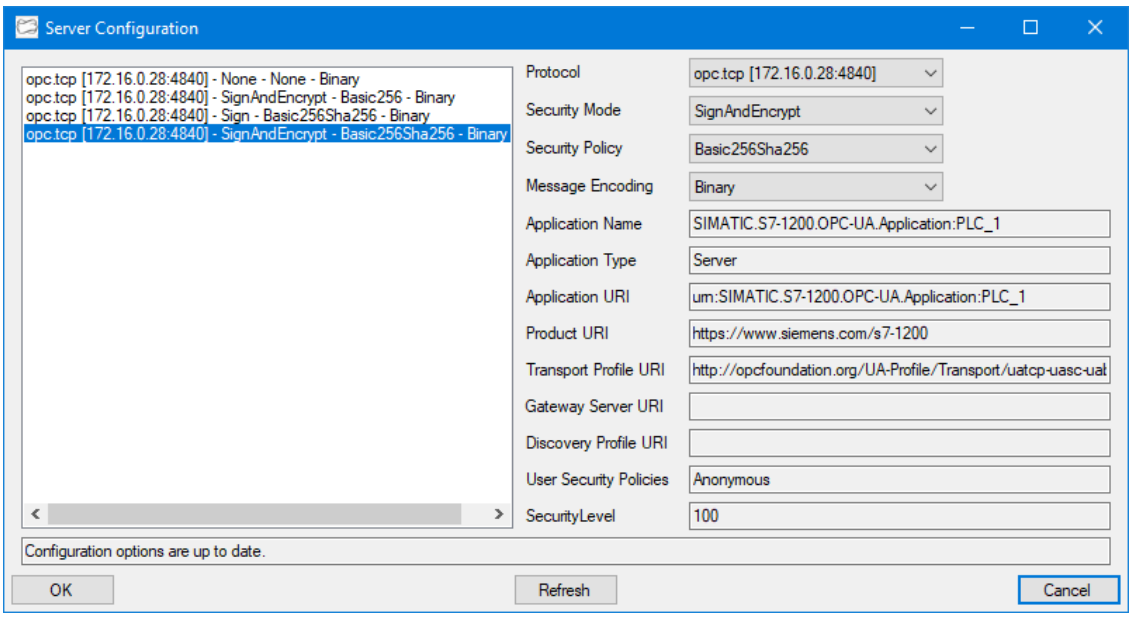
Parametri "**Server settings**".

## Use Security

La flag "Use Security" specifica la preferenza sull'utilizzo o meno della sicurezza; in base alla preferenza e alle proprie capacità il server seleziona e propone un Endpoint di connessione compatibile.
Nel caso di server che non supportano la sicurezza la disabilitazione della flag dovrebbe agevolare la selezione dell'Endpoint senza sicurezza con le caratteristiche "Security mode" e "Security Policy" impostate a "None".

## Configure Security

L'opzione visualizza la finestra "Server Configuration":




Nell'elenco a sinistra sono visualizzate le modalità di connessione (i cosiddetti "Endpoint") disponibili sul server **(*)**; nella parte a destra sono visualizzate le informazioni di dettaglio.

La modalità può essere selezionata direttamente nell'elenco di sinistra o impostando singolarmente i parametri nell'area a destra:

- Protocol
- Security Mode
- Security Policy
- Message Encoding

Confermando con "OK" si ritorna alla finestra principale delle proprietà dove verranno visualizzate le informazioni aggiornate in base alla modalità selezionata.

**Note**

**(*)** Se l'accesso a questa finestra avviene in condizioni di off-line, quando cioè non è attivo un collegamento con il server, non sarà possibile apportare modifiche.
Le informazioni visualizzate saranno esclusivamente quelle correntemente memorizzate nella configurazione (nell'elenco a sinistra sarà presente una sola riga). Nella parte inferiore della finestra verrà visualizzato un messaggio di avviso.

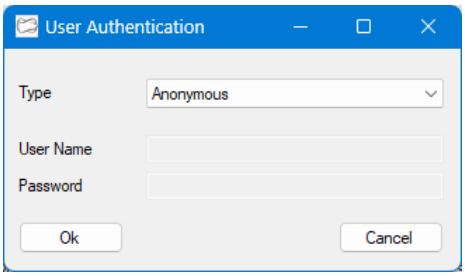
**Disable host name check**

L'impostazione consente di disabilitare il controllo dell'hostname presente nel certificato del server.

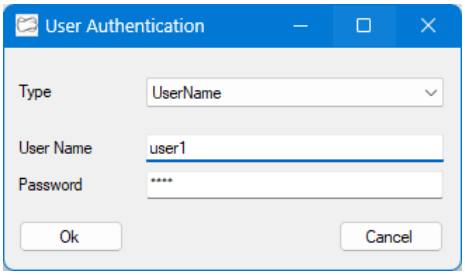
L'impostazione può essere necessaria nel caso in cui nella barra di stato si visualizza il messaggio "Server not connected – BadCertificateHostNameInvalid"; per ulteriori informazioni fare riferimento al paragrafo "Troubleshooting" a pag. 53.

**Configure User Authentication**

L'opzione propone la finestra "User Authentication":




Sono supportati i livelli "Anonymous" e "Username"; l'elenco consente di scegliere tra i livelli supportati dal server.
Nel caso del livello "Anonymous" non sono richieste altre informazioni, mentre con il livello "Username" è necessario inserire "User Name" e "Password".




### 8.5.2.3 Cancellazione di un server

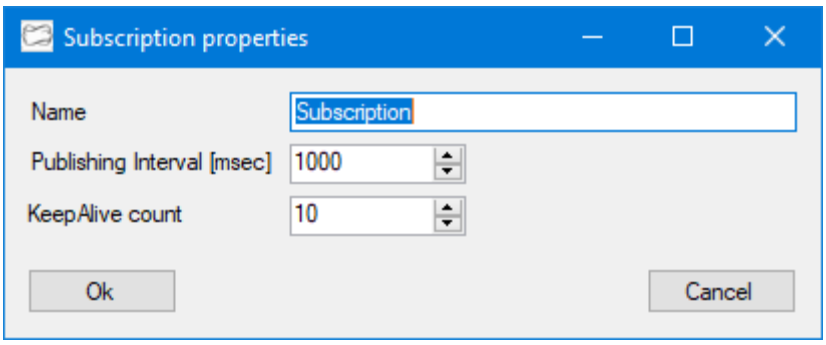
Prima di eseguire l'operazione di cancellazione di un server viene richiesta la conferma.

### 8.5.3 Configurazione di una subscription

Dall'albero della configurazione è possibile accedere ai comandi per creare una nuova subscription oppure modificare o cancellare una subscription esistente.

#### 8.5.3.1 Creazione e modifica di una subscription

Dopo il comando di creazione o modifica di una subscription si visualizzerà la finestra delle relative proprietà:




| Name |
|---|
| Nome della subscription. |
| **Publishing Interval** |
| Tempo richiesto per l'aggiornamento degli item (frequenza di invio delle notifiche da parte del server).<br>**Limiti** ▪ il valore è espresso in millisecondi e il range è compreso tra 0 e 1000000000 |
| **KeepAlive Count** |
| Numero di cicli di Publishing 'vuoti' (che non hanno generato notifiche) consecutivi che fanno scattare l'invio di un messaggio di KeepAlive al client.<br>**Limiti** ▪ il range del valore è compreso tra 0 e 1000 |

Il valore assegnato ai parametri è da considerare come 'indicativo/desiderato'; il server può applicare degli aggiustamenti. Se impostato a 0 il server adotterà il valore minimo in base alle proprie capacità.

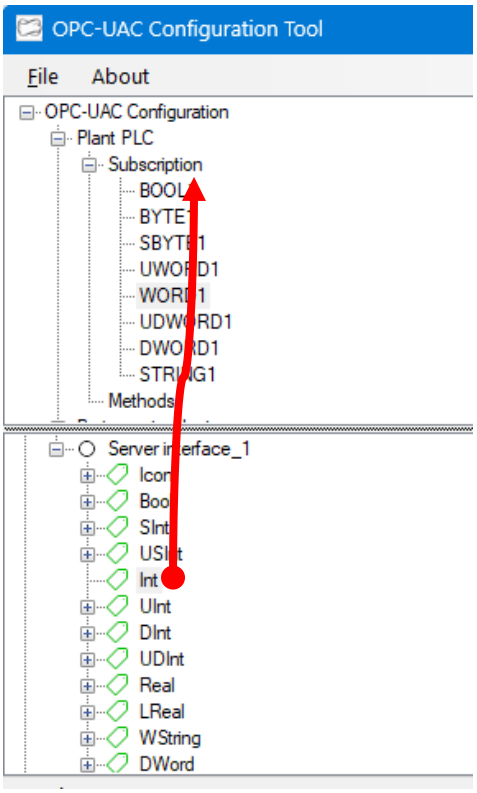
#### 8.5.3.2 Cancellazione di una subscription

Prima di eseguire l'operazione di cancellazione di una subscription viene richiesta la conferma.

## 8.5.4 Configurazione di un item

### 8.5.4.1 Creazione di un item

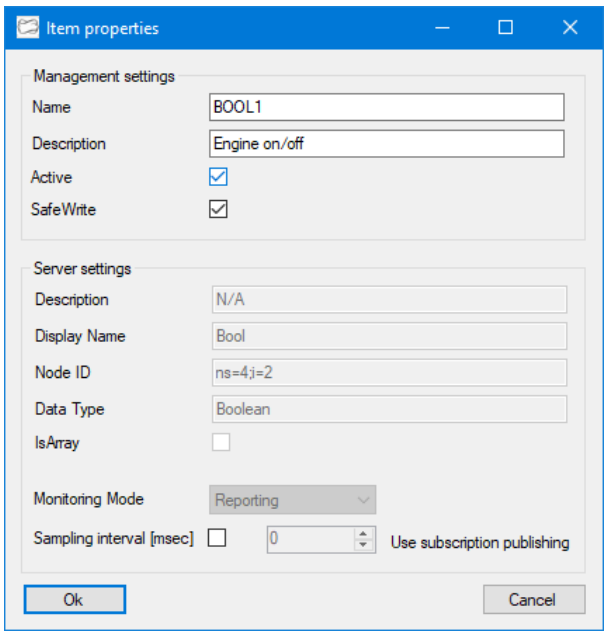
L'aggiunta di un item alla configurazione avviene tramite un'operazione di drag&drop trascinando la risorsa di tipo "Variable" selezionata nell'albero delle risorse del server nella subscription di destinazione (il punto di drop può essere il nodo subscription o uno degli item già contenuti). Gli elementi di tipo "Variable" sono identificati dall'icona .




Il drop è consentito solo all'interno di una subscription assegnata al server sorgente; se l'operazione va a buon fine viene immediatamente visualizzata la finestra delle proprietà dove è possibile completare le impostazioni dell'item.

### 8.5.4.2 Modifica di un item

Dopo il comando di creazione o modifica di un item si visualizza la finestra delle proprietà:




Le proprietà sono suddivise in due gruppi:

- "Management settings": parametri che regolano alcuni aspetti di gestione dell'item
- "Server settings": parametri che riportano alcune delle proprietà dell'item così come dichiarato nel server e che regolano il suo funzionamento. Di particolare importanza è la proprietà "Node ID" che identifica in modo univoco l'item nella basedati del server.

Parametri "**Management settings**".

| Name |
|---|
| Nome da utilizzare per identificare l'item: deve essere unico in tutta la configurazione. |
| **Description** |
| Descrizione libera da associare all'item. |
| **Active** |
| Abilitazione dell'item. |

**Note** Gli item non attivi sono conservati nella configurazione ma non sono gestiti nel runtime; nel caso di utilizzo nelle variabili verrà generato un codice di errore specifico. Per un dettaglio sui codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51

### Safe Write

Attiva la modalità di scrittura "sicura".

Di massima l'operazione di scrittura viene effettuata in modo bloccante per il tempo necessario a garantire la consegna del valore alla periferica o determinarne l'eventuale impossibilità.
Nel caso di una sistema strutturato su più livelli il server potrebbe però consegnare il valore a un sistema intermedio che completa l'operazione in modalità asincrona (in questo caso il server informa il client tramite uno status specifico); può quindi succedere che un'eventuale lettura ravvicinata possa restituire un valore non ancora aggiornato.
Attivando la modalità "Safe Write", nella situazione appena descritta, dopo aver inviato al server la richiesta di scrittura il driver esegue delle letture fino a quando il valore letto coincide con quello appena impostato o al raggiungimento del valore di timeout (parametro "Safe Write timeout" impostato nell'elemento server). Nel caso di terminazione per timeout verrà generato un errore.
Per un dettaglio sui codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51

**Note** Questa modalità può essere utilizzata solo se il dato viene scritto esclusivamente da un unico applicativo client.

Parametri "**Server settings**"

### Sampling interval

Indica il tempo di campionamento dell'item nel server (acquisizione dalla periferica).

Se la check è disabilitata (condizione standard di default) il parametro viene assegnato pari al valore del "Publishing Interval" della subscription.

Abilitando la check è invece possibile impostare un tempo personalizzato.

Il valore assegnato al parametro è da considerare come 'indicativo/desiderato'; il server può applicare degli aggiustamenti. Se impostato a 0 il server adotterà il valore minimo in base alle proprie capacità.

**Limiti** ▪ il valore è espresso in millisecondi e il range è compreso tra 0 e 1000000000

Alla conferma con "OK" viene eseguita una verifica sul campo "Name" e nel caso di errori la finestra delle proprietà rimane attiva; possono essere segnalati i seguenti errori:

| Errore | Descrizione |
|---|---|
| The item name can not be empty | Il campo Name non può essere vuoto |
| The name xxx is already used | L'item citato è già presente nella configurazione |

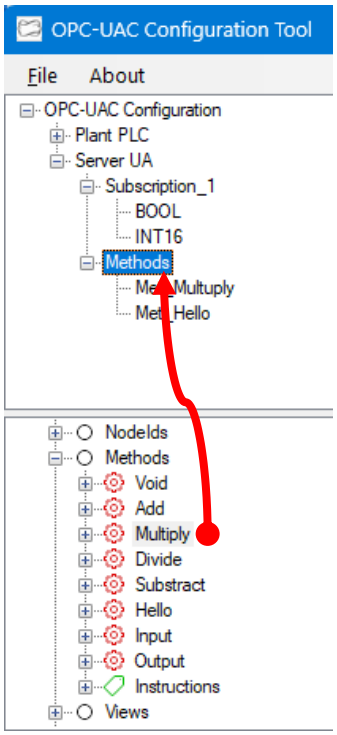
## 8.5.4.3 Cancellazione di un item

Prima di eseguire l'operazione di cancellazione di un item viene richiesta la conferma.

## 8.5.5 Configurazione di un metodo

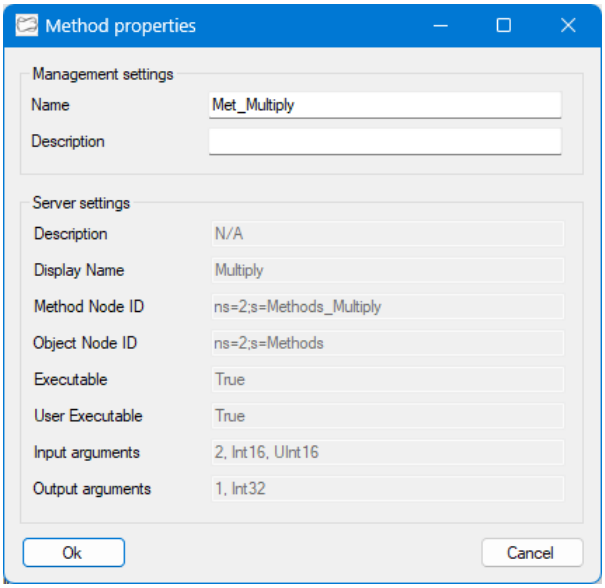
### 8.5.5.1 Creazione di un metodo

L'aggiunta di un metodo alla configurazione avviene tramite un'operazione di drag&drop trascinando la risorsa di tipo "Method" selezionata nell'albero delle risorse del server nel gruppo predefinito "Methods" (il punto di drop può essere il nodo "Methods" o uno dei metodi già contenuti).
Gli elementi di tipo "Method" sono identificati dall'icona .




### 8.5.5.2 Modifica di un metodo

Dopo il comando di creazione o modifica di un metodo si visualizza la finestra delle proprietà:



| Name |
| --- |
| Nome da utilizzare nell'indirizzo di una variabile SUPER-FLASH per identificare il metodo. |
| **Description** |
| Descrizione libera da associare al metodo. |

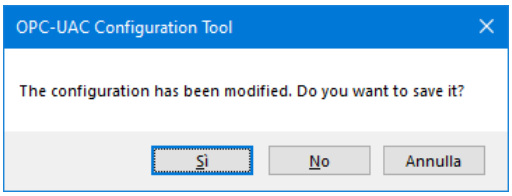
**Note**
Un metodo fa sempre riferimento a un oggetto; in fase di creazione si considera come oggetto di riferimento il nodo contenitore del metodo stesso (indicato come "Object Node ID").

### 8.5.5.3 Cancellazione di un metodo

Prima di eseguire l'operazione di cancellazione di un item viene richiesta la conferma.

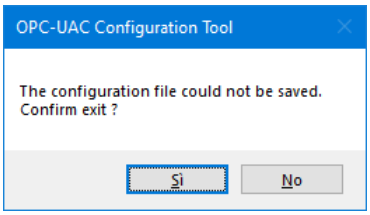
## 8.5.6 Salvataggio della configurazione

Questo comando memorizza su disco la configurazione nel file "OPCUAC.XML".
All'uscita dal configuratore, se la configurazione è stata modificata ma non salvata, verrà fatta la seguente richiesta:




- **"Sì":** salva la configurazione
- **"No" :** non salva la configurazione perdendo tutte le modifiche
- **"Annulla" :** annulla il comando di uscita

Se il salvataggio del file fallisce, es. per un problema di accesso al file, viene visualizzata una richiesta di conferma per proseguire con la chiusura dell'applicazione.



## 8.6 Descrizione delle funzioni per categoria

Le funzioni possono essere raggruppate nelle seguenti categorie:

- inizializzazione e terminazione
- data access (lettura, scrittura e notifica valori)
- chiamata metodi
- informazioni

### 8.6.1 Inizializzazione e terminazione

| | |
|---|---|
| OpcuacInit | Inizializza la libreria caricando il file di configurazione dell'applicazione, verificando i certificati e caricando la basedati degli item. |
| OpcuacSetItemNotifier | Imposta la funzione di callback per la notifica delle variazioni dei valori degli item. |
| OpcuacStart | Avvia la connessione ai server e l'aggiornamento degli item. |
| OpcuacStop | Chiude la connessione ai server, interrompe l'aggiornamento degli item e scarica la configurazione. |

Le funzioni devono essere chiamate nel seguente ordine:

1. OpcuacSetItemNotifier (opzionale)
2. OpcuacInit
3. OpcuacStart
4. …
5. OpcuacStop

**Esempio**

C/C++

```
#include "OPCUAC.H"

static void Notify(OPCUACItemNotification *pNotif) ;

void InitOPCUAC()
{
	OpcuacSetItemNotifier(Notify) ;
	unsigned int err = OpcuacInit() ;

	if (err == ERR_None)
		OpcuacStart() ;

	InitMessage(err) ;
}

void Notify(OPCUACItemNotification *pNotif)
{
	//Manage Notification
}

void TerminateOPCUAC()
{
	OpcuacStop() ;
}
```

C#

```
private void InitOPCUAC()
{
	OPCUACItemNotifyDelegate del = Notify;
	OpcuacInterface.OpcuacSetItemNotifier(del);

	uint err = OpcuacInterface.OpcuacInit();

	if (err == OpcuacInterface.Opcuac_ERR_None)
		OpcuacInterface.OpcuacStart();

	InitMessage(err) ;

}

private void Notify(OPCUACItemNotification notif)
{
	//Manage Notification
}

private void TerminateOPCUAC()
{
	OpcuacInterface.OpcuacStop();
}
```

### 8.6.2 Gestione Data Access

| | |
|---|---|
| OpcuacGetValue | Ottiene l'ultimo valore aggiornato di un item dalla cache interna della libreria. |
| OpcuacReadValue | Legge il valore di un item dal server. |
| OpcuacWriteValue | Scrive il valore di un item. |
| OpcuacFreeReadValue (*) | Rilascia la memoria di gestione della lettura dell'item. |
| OpcuacFreeNotification (*) | Rilascia la memoria di gestione della notifica dell'item. |

(*) Solo per l'interfaccia C/C++

#### Esempio

C/C++

```
#include "OPCUAC.H"

void unsigned int ReadWriteItem(wchar_t *name)
{
	OPCUACItemReadInfo tmpr ;

	int numval = 0 ; //all
	int index = 0 ;
	unsigned int err ;

	::ZeroMemory(&tmpr, sizeof(tmpr)) ;
	if ((err = OpcuacGetValue(name, numval, index, &tmpr)) == ERR_None)
		{
		OPCUACItemWriteInfo tmpw ;

		::ZeroMemory(&tmpw, sizeof(tmpw)) ;
		tmpw.Value = tmpr.Value ;
		tmpw.ValueSize = tmpr.ValueSize ;

		err = OpcuacWriteValue(name, tmpr.ValueNum, 0, &tmpw) ;
		}
	OpcuacFreeReadValue(&tmpr) ;

}
…

void Notify(OPCUACItemNotification *pNotif)
{
	if (pNotif != NULL)
	{
		UpdateItem(pNotif->ItemName, &pNotif->Value) ;
		OpcuacFreeItemNotification(pNotif) ;
	}
}
```

C#

```
private unsigned int ReadWriteItem(string name)
{
	OPCUACItemReadInfo tmpr;
	uint err;

	OPCUACItemReadInfo tmpr = new OPCUACItemReadInfo();
	uint err;

	string name = m_ItemName.Text;
	name.TrimEnd();

	if ((err = OpcuacInterface.OpcuacGetValue(name, 0, 0, tmpr)) ==
		OpcuacInterface.Opcuac_ERR_None)
	{
		OPCUACItemWriteInfo tmpw = new OPCUACItemWriteInfo();

		tmpw.Value = tmpr.Value;
		err = OpcuacInterface.OpcuacWriteValue(name, tmpr.ValueNum, 0, tmpw);
	}

	return (err) ;
}

private void OnNotify(OPCUACItemNotification notif)
{
	UpdateItem(notif.ItemName, notif.Value) ;
}
```

## 8.6.3 Gestione chiamata a metodi

| OpcuacCallMethod (*) | Esegue un metodo |
|---|---|

(*) Disponibile solo nel'interfaccia C#.

### Esempio

C#

```
private unsigned int CallMethodMultiply()
{
	OPCUACMethodInfo met = new OPCUACMethodInfo();
	object[] input = new object[2];

	System.Int16 p1 = 10;
	System.UInt16 p2 = 5;
	input[0] = p1;
	input[1] = p2;
	met.InputPar = input;
	uint err = OpcuacInterface.OpcuacCallMethod("Multiply", met);

	if (err == 0 && met.OutputPar != null)
	{
		string msgresult = String.Format("{0} x {1} = {2}", met.InputPar[0],
		met.InputPar[1],
		met.OutputPar[0]);
		MessageBox.Show(msgresult, "Method Multiply result");
	}
	else
	{
		string msgresult = String.Format("Method result: {0:X} [{1}]", err,
		OpcuacInterface.OpcuacErrorDescription(err));
		MessageBox.Show(msgresult, "Method Multiply result");
	}
	return (err) ;
}
```

### 8.6.4 Informazione

| | |
|---|---|
| OpcuacVersion | Fornisce la versione della libreria |
| OpcuacProtection | Fornisce la modalità di funzionamento della libreria (demo / full) |
| OpcuacErrorDescription | Fornisce la stringa descrittiva associata al codice di errore/status (in lingua inglese). |
| OpcuacValueStatusInfo | Fornisce lo stato di usabilità del dato (Good/Uncertain/Bad) |

**Esempio**

C/C++

```
#include "OPCUAC.H"

void InitMessage(unsigned int st)
{
	CString msg, msgst, msgver, msglic, strst;

	unsigned int Ver = OpcuacVersion();
	msgver.Format(L"Version: %d.%02d", Version / 100, Version % 100);

	unsigned int Prot = OpcuacProtection();
	msglic = Prot != 0 ? L"Licence: Ok" : L"Licence Demo";

	strst = OpcuacErrorDescription(st);
	msgst.Format(L"OPCUAC Initialization: [%x = %s]", st, strst);

	msg.Format(L"%s\r\n%s\r\n%s", msgver, msglic, msgst);

	MessageBox(msg, L"OPCUAC demo", MB_OK);
}

void DisplayValueStatus(OPCUACItemReadInfo *info)
{
	unsigned int stcat = OpcuacValueStatusInfo(info->ValueStatus) ;
	wchar_t *str = L"Unknown";

	if (stcat ==_DS_Good)
		str = L"Good" ;
	else if (stcat == DS_Uncertain)
		str = L"Uncertain" ;
	else if (stcat == DS_Bad)
		str = L"Bad" ;

	m_Quality = str ;
}
```

C#

```
private void InitMessage(uint st)
{
	String msg, msgst, msgver, msglic, strst;

	UInt32 Version = OpcuacInterface.OpcuacVersion();
	msgver = String.Format("Version: {0}.{1}", Version / 100, Version % 100);

	UInt32 Prot = OpcuacInterface.OpcuacProtection();
	msglic = Prot != 0 ? "Licence: Ok" : "Licence Demo";

	strst = OpcuacInterface.OpcuacErrorDescription(st);
	msgst = String.Format("OPCUAC Initialization: [{0:X} = {1}]",st, strst);

	msg = String.Format("{0}\r\n{1}\r\n{2}", msgver, msglic, msgst);

	MessageBox.Show(msg, this.Text);

}

private void DisplayDataStatus(OPCUACItemReadInfo info)
{
	uint stcat = OpcuacInterface.OpcuacValueStatusInfo(info.ValueStatus) ;
	string str = "Unknown";

	if (stcat == OpcuacInterface.Opcuac_DS_Good)
		str = "Good" ;
	else if (stcat == OpcuacInterface.Opcuac_DS_Uncertain)
		str = "Uncertain" ;
	else if (stcat == OpcuacInterface.Opcuac_DS_Bad)
		str = "Bad" ;

	m_Quality.Text = str ;
}
```

# 9 Funzioni in ordine alfabetico

In questo capitolo vengono descritte una ad una, in ordine alfabetico, tutte le funzioni disponibili. La scheda riporta le seguenti informazioni:

| | |
|---|---|
| Nome | Nome della funzione |
| Versioni obsolete | Eventuali nomi obsoleti della stessa funzione |
| Interfaccia completa | • Tipo del valore di ritorno della funzione<br>• Nome della funzione<br>• Parametri in ordine e loro tipo |
| Parametri | Significato sintetico di ciascun parametro |
| Descrizione | Scopo della funzione |
| Osservazioni | Note su alcuni comportamenti particolari e su alcune attenzioni da osservare nell'utilizzo della funzione |
| Valore di ritorno | Significato del valore di ritorno della funzione |
| Tabella identificativa | Vedi seguito |

La scheda della funzione è completata una tabella identificativa che riporta le seguenti informazioni:

| | |
|---|---|
| Stato | • Ok: la funzione è attiva e utilizzabile<br>• Obsoleto: la funzione non è da utilizzare |
| Prodotto | Prodotto di appartenenza |
| Categoria | Categoria di funzioni di appartenenza. |
| Referenze | Elenco delle funzioni della stessa categoria. |
| Esempi | Esempi di utilizzo della categoria. |
| In OPC_UAC | Versione di OPC-UAC in cui la funzione è stata inserita.<br>Se la funzione è obsoleta, viene citata la versione di OPC-UAC in cui è stata dichiarata obsoleta. |

## 9.1 OpcuacCallMethod

C#

**unsigned int OpcuacInterface.OpcuacCallMethod(string item, OPCUACMethodInfo pdBuf)**

La funzione *OpcuacCallMethod* esegue la chiamata del metodo associato all'item specificato.

**item** Identificativo alfanumerico dell'item associato al metodo da eseguire (attributo "Name" del metodo).

**pdbuf** Struttura che contiene gli argomenti di input e output e il codice di dettaglio dell'esito della chiamata.

**Osservazioni** I parametri di input e output sono gestiti come array di System::Object; per i dettagli su come gestire un oggetto System::Object rispetto al tipo di dato OPC UA fare riferimento alle osservazioni della funzione *OpcuacGetValue* (note riferite a C#).

Se l'esito dell'operazione è l'errore specifico di esecuzione fallita è possibile consultare il codice di dettaglio **ExecutionStatus** contenuto nella struttura **pdbuf**; i possibili valori sono elencati al paragrafo "Status dell'esecuzione di un metodo" a pag. 49.

**Valore di ritorno** L'esito dell'operazione; 0 (zero) se l'operazione è andata a buon fine altrimenti un codice di errore.
Per l'elenco dei possibili codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione chiamata a metodi a pag. 24 |
| Referenze | A pag. 24 |
| Esempi | A pag. 24 |
| In OPC-UAC | Dalla versione 2.0 |

## 9.2 OpcuacErrorDescription

C/C++
**wchar_t*  OpcuacErrorDescription(unsigned int err)**

C#
**string  OpcuacInterface.OpcuacErrorDescription(uint err)**

La funzione *OpcuacErrorDescription* restituisce la stringa descrittiva associata al codice di errore specificato.

| | |
|---|---|
| **err** | Codice di errore restituito dalle funzioni di lettura e scrittura dell'item |
| **Osservazioni** | C/C++: la funzione utilizza un'area static; ogni chiamata sovrascrive il risultato della precedente. |
| **Valore di ritorno** | La stringa descrittiva dell'errore. |

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di informazione pag. 25 |
| Referenze | A pag. 25 |
| Esempi | A pag. 25 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.3 OpcuacFreeItemNotification

C/C++

**void OpcuacFreeItemNotification(OPCUACItemNotificationInfo* notif)**

La funzione *OpcuacFreeItemNotification* libera la memoria utilizzata dalla struttura di tipo OPCUACItemNotificationInfo per la gestione della notifica di variazione del valore.

| | |
|---|---|
| **notif** | Struttura ricevuta tramite la funzione di callback di notifica variazione del valore. |

| | |
|---|---|
| **Osservazioni** | La funzione esiste solo nell'interfaccia C/C++. |
| **Valore di ritorno** | Nessuno. |

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione valori pag. 23 |
| Referenze | A pag. 23 |
| Esempi | A pag. 23 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.4 OpcuacFreeReadValue

C/C++

**void OpcuacFreeReadValue(OPCUACItemReadInfo* read)**

La funzione *OpcuacFreeReadValue* libera la memoria relativa al valore dell'item contenuto nella struttura di tipo OPCUACItemReadInfo.

| | |
|---|---|
| **read** | Struttura contenente il valore dell'item restituito dalla funzione di lettura. |
| **Osservazioni** | La funzione esiste solo nell'interfaccia C/C++. |
| **Valore di ritorno** | Nessuno. |

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione valori pag. 23 |
| Referenze | A pag. 23 |
| Esempi | A pag. 23 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.5 OpcuacGetValue

C/C++
**unsigned int OpcuacGetValue(wchar_t* item, int numval, int index, OPCUACItemReadInfo* pdBuf)**

C#
**uint OpcuacInterface.OpcuacGetValue(string item, int numval, int index, OPCUACItemReadInfo pdBuf)**

La funzione *OpcuacGetValue* restituisce l'ultimo valore disponibile dell'item memorizzato nella cache interna della libreria.
Se l'attivazione del servizio di subscription non è andato a buon fine il valore viene letto dal server.

| | |
|---|---|
| **item** | Identificativo alfanumerico dell'item da leggere (attributo "Name" del'item) |
| **numval** | Numero di valori da restituire:<br><=0 restituisce l'intero valore (singolo o array)<br>>0 restituisce il numero di valori richiesti |
| **index** | Indice del primo valore da restituire, i valori validi sono:<br>0 per valori singoli ("Scalar")<br>0/n-1 per valori array, dove n è il numero di elementi nell'array |
| **pdbuf** | Struttura in cui caricare il valore e le informazioni associate. |

**Osservazioni**

Le versioni gestita e non gestita della struttura OPCUACItemReadInfo differiscono solo per la modalità di gestione del valore.

**ValueRank** indica se il valore è singolo o è un array.
I valori che può assumere sono quelli indicati nelle costanti RV_xxx.

**ValueType** indica il DataType del valore (Boolean, Byte, ecc.).
I valori che può assumere sono quelli indicati nelle costanti DT_xxx.

**ValueNum** 1 se il valore è singolo oppure il numero totale degli elementi nell'array.

**ValueStatus** lo status associato al valore.
Per verificare la validità del valore è possibile utilizzare la funzione *OpcuacValueStatusInfo.*

**ServerTS** il server timestamp associato al valore.

**SourceTS** il source timestamp associato al valore.

**Value** il valore.

C#: nella struttura gestita, **Value** è costituito da un oggetto di tipo System::Object. Il tipo di oggetto dipende dalle caratteristiche di **ValueRank** e **ValueType** dell'item.
Es:

| | |
|---|---|
| Valore singolo, Boolean: | bool |
| Valore array, Boolean: | bool[] |
| Valore singolo, Byte: | byte |
| Valore array, Byte: | byte[] |

| | |
|---|---|
| Valore singolo, SByte: | sbyte |
| Valore array, SByte: | sbyte[] |
| Valore singolo, UInt16: | ushort |
| Valore array, UInt16: | ushort[] |
| Valore singolo, Int16: | short |
| Valore array, Int16: | short[] |
| Valore singolo, UInt32: | uint |
| Valore array, UInt32: | uint[] |
| Valore singolo, Int32: | int |
| Valore array, Int32: | int[] |
| Valore singolo, Float: | float |
| Valore array, Float: | float[] |
| Valore singolo, Double: | double |
| Valore array, Double: | double[] |
| Valore singolo, Bytestring: | byte[] |
| Valore array, Bytestring: | byte[][] |
| Valore singolo, String: | string |
| Valore array, String: | string[] |
| Valore singolo, LocalizedText: | string |
| Valore array, LocalizedText: | string[] |

C/C++: nella struttura non gestita, **Value** è un generico puntatore void* che va interpretato sulla base delle altre informazioni, come descritto di seguito.

Il numero di valori contenuti in **Value** è **ValueNum** se la richiesta è stata fatta per l'intero item (**numval** <= 0) altrimenti è il numero richiesto **numval**.

Il tipo del valore è dato dal campo **ValueType**.

Tipi Bytestring (sequenza di byte), String o LocalizedText: **Value** contiene delle strutture di tipo OPCUACStringData; una per ogni valore restituito.
La struttura OPCUACStringData contiene il campo **StrValue** e l'informazione di dimensione **StrValueSize** espressa in byte.
Se il dato è di tipo String o LocalizedText **StrValue** va interpretato come un array di wchar_t con dimensione **StrValueSize/2**.
Se il dato è vuoto **StrValue** è NULL e **StrValueSize** è 0.
Attenzione! il valore di tipo String o LocalizedText non include il carattere terminatore.

Tutti gli altri tipi: **Value** è un buffer di byte che contiene tutti i valori; il campo **ValueSize** indica la dimensione totale.

Per i valori singoli la dimensione è:

| | |
|---|---|
| Boolean, Byte, Sbyte | **ValueSize** = 1 |
| Int16,UInt16 | **ValueSize** = 2 |
| Int32, UInt32, Single (float) | **ValueSize** = 4 |
| Double | **ValueSize** = 8 |
| ByteString,String,LocalizedText | **ValueSize**= sizeof(OPCUACStringData) |

Per i valori di tipo array la dimensione è:
**ValueSize** = *dimensione singolo * numero valori*
L'unica eccezione è per il tipo array di Boolean in cui i bit sono compattati e quindi la dimensione totale corrisponde a:
**ValueSize** = *(numero valori + 7) / 8*

**Attenzione**: al termine dell'utilizzo della struttura non gestita è necessario passarla alla funzione *OpcuacFreeReadValue* per liberare la memoria allocata dalla libreria per il campo **Value.**

**Valore di ritorno**

L'esito della lettura; 0 (zero) se l'operazione è andata a buon fine altrimenti un codice di errore.
Per l'elenco dei possibili codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione valori pag. 23 |
| Referenze | A pag. 23 |
| Esempi | A pag. 23 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.6 OpcuacInit

C/C++
**unsigned int OpcuacInit()**

C#
**uint OpcuacInterface.OpcuacInit()**

La funzione *OpcuacInit* esegue l'inizializzazione della libreria.
L'inizializzazione è composta da queste operazioni:

- caricamento del file di configurazione dell'applicazione "OPCUAC.Config.xml"
- caricamento (eventuale generazione o rinnovo) dell'"Application Certificate"
- caricamento della basedati dal file "OPCUAC.XML"

**Osservazioni** Nessuna.

**Valore di ritorno** L'esito della procedura di inizializzazione; 0 se l'operazione è andata a buon fine altrimenti un codice di errore.
Per l'elenco dei possibili codici di errore fare riferimento al paragrafo "Errori generali" a pag. 50.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di inizializzazione pag. 21 |
| Referenze | A pag. 21 |
| Esempi | A pag. 21 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.7 OpcuacProtection

C/C++
**unsigned int OpcuacProtection()**

C#
**uint OpcuacInterface.Protection()**

La funzione *OpcuacProtection* restituisce la modalità di funzionamento della libreria rispetto la presenza del sistema di protezione (chiave hardware o codice di attivazione software).

**Osservazioni**

Nel caso di assenza della protezione la libreria funziona in modalità demo per un tempo di 15 minuti. Trascorso il tempo di demo le funzioni di lettura e scrittura restituiscono il codice di errore FFFFFF92H e il meccanismo di notifica viene interrotto.

**Valore di ritorno**

Valore diverso da 0 (zero): la protezione è stata rilevata e la libreria funziona in modalità normale.
Valore 0 (zero): la protezione NON è stata rilevata e la libreria funziona in modalità demo.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di informazione pag. 25 |
| Referenze | A pag. 25 |
| Esempi | A pag. 25 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.8 OpcuacReadValue

C/C++

**unsigned int OpcuacReadValue(wchar_t* item, int numval, int index, OPCUACItemReadInfo* pdBuf)**

C#

**uint OpcuacInterface.OpcuacReadValue(string item, int numval, int index, OPCUACItemReadInfo pdBuf)**

La funzione *OpcuacReadValue* legge il valore dell'item dal server.

| | |
|---|---|
| **item** | Identificativo alfanumerico dell'item da leggere (attributo "Name" del'item) |
| **numval** | Numero di valori da restituire:<br><=0 restituisce l'intero valore (singolo o array)<br>>0 restituisce il numero di valori richiesti |
| **index** | Indice del primo valore da restituire, i valori validi sono:<br>0 per valori singoli ("Scalar")<br>0/n-1 per valori array, dove n è il numero di elementi nell'array |
| **pdbuf** | Struttura in cui caricare il valore e le informazioni associate. |
| **Osservazioni** | Fare riferimento alle osservazioni della funzione *OpcuacGetValue.* |
| **Valore di ritorno** | L'esito della lettura; 0 (zero) se l'operazione è andata a buon fine altrimenti un codice di errore.<br>Per l'elenco dei possibili codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51. |

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione valori pag. 23 |
| Referenze | A pag. 23 |
| Esempi | A pag. 23 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.9 OpcuacSetItemNotifier

C/C++

**void OpcuacSetItemNotifier(void (*pfun)(OPCUACItemNotificationInfo* fun))**

C#

**void OpcuacInterface.OpcuacSetItemNotifier(OPCUACItemNotifyDelegate fun)**

La funzione *OpcuacSetItemNotifier* imposta la funzione di callback che verrà utilizzata per notificare l'avvenuto aggiornamento del valore di un item.

**fun** Funzione di gestione delle notifiche di variazione del valore degli item

**Osservazioni** La struttura OPCUACItemNotificationInfo passata alla funzione di callback contiene i seguenti campi

**ItemName** identificativo dell'item che è stato aggiornato.

**Value** valore aggiornato dell'item. Il campo è costituito da una struttura di tipo OPCUACItemReadInfo descritta nella funzione *OpcuacGetValue.*

La funzione deve essere impostata prima di chiamare la funzione di inizializzazione *OpcuacInit.*

**Attenzione**: al termine dell'utilizzo della struttura non gestita è necessario passare il puntatore alla funzione *OpcuacFreeNotification* per liberare la memoria allocata dalla libreria.

**Valore di ritorno** Nessuno.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di inizializzazione pag. 21 |
| Referenze | A pag. 21 |
| Esempi | A pag. 21 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.10 OpcuacStart

C/C++
**void OpcuacStart()**

C#
**void OpcuacInterface.OpcuacStart()**

La funzione *OpcuacStart* avvia le connessioni ai server e il meccanismo di aggiornamento e notifica di variazione dei valori degli item.

**Osservazioni** Nessuna.

**Valore di ritorno** Nessuno.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di inizializzazione pag. 21 |
| Referenze | A pag. 21 |
| Esempi | A pag. 21 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.11 OpcuacStop

C/C++
**void OpcuacStop()**

C#
**void OpcuacInterface.OpcuacStop()**

La funzione *OpcuacStop* interrompe le connessioni ai server e il meccanismo di aggiornamento e notifica di variazione dei valori degli item.

**Osservazioni** Nessuna.

**Valore di ritorno** Nessuno.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di inizializzazione pag. 21 |
| Referenze | A pag. 21 |
| Esempi | A pag. 21 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.12 OpcuacValueStatusInfo

C/C++
**unsigned int OpcuacValueStatusInfo(unsigned int status)**

C#
**uint OpcuacInterface.OpcuacValueStatusInfo(uint status)**

La funzione *OpcuacValueStatusInfo* restituisce l'indicatore di usabilità del valore, letto o notificato, di un item.

| | |
|---|---|
| **status** | Status associato al valore dell'item. |
| **Osservazioni** | I possibili valori che **status** può assumere sono elencati al paragrafo "Status del valore" a pag. 48. |
| **Valore di ritorno** | L'indicatore di usabilità del valore.<br>L'indicatore può assumere tre valori identificati dalle seguenti costanti:<br>• DS_Good<br>• DS_Uncertain<br>• DS_Bad |

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di informazione pag. 25 |
| Referenze | A pag. 25 |
| Esempi | A pag. 25 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.13 OpcuacVersion

C/C++
**unsigned int OpcuacVersion()**

C#
**uint OpcuacInterface.Version()**

La funzione *OpcuacVersion* restituisce, sotto forma di valore numerico, la versione di OPC-UAC.

**Osservazioni** La versione è espressa nel seguente modo:
- numero di versione principale * 100
- più numero di versione secondaria

Esempio: versione 120 = 1.20

**Valore di ritorno** La versione di OPC-UAC.

| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di informazione pag. 25 |
| Referenze | A pag. 25 |
| Esempi | A pag. 25 |
| In OPC-UAC | Dalla versione 1.0 |

## 9.14 OpcuacWriteValue

C/C++

**unsigned int OpcuacWriteValue(wchar_t* item, int numval, int index, OPCUACItemWriteInfo* pdBuf)**

C#

**uint OpcuacInterface.OpcuacWriteValue(string item, int numval, int index, OPCUACItemWriteInfo pdBuf)**

La funzione *OpcuacWriteValue* scrive il valore di un item.

| | |
|---|---|
| **item** | Identificativo alfanumerico dell'item da scrivere (attributo "Name" del'item) |
| **numval** | Numero di valori da scrivere<br>1 per valori singoli ("Scalar")<br>1/n per valori array: numero di elementi da modificare, , dove n è il numero di elementi attualmente contenuti nel'array<br>-1 per valori array: innesca la scrittura dell'intero array (per array dinamici) |
| **index** | Indice del primo valore da scrivere, i valori validi sono:<br>0 per valori singoli ("Scalar")<br>0/n-1 per valori array, dove n è il numero di elementi attualmente contenuti nel'array |
| **pdbuf** | Struttura contenente il valore da scrivere. |
| **Osservazioni** | Utilizzo del parametro **numval** nel caso di valoru array:<br>• per modificare uno o più elementi del valore array attuale assegnare a numval un valore >= 1<br>• per scrivere l'intero array assegnare a **numval** il valore -1; questa modalità può essere utilizzata per gli array dinamici in modo da sovrascrivere completamente il valore (es. nel caso di variazione del numero di elementi).<br><br>Campo **Value** della struttura OPCUACItemWriteInfo: contiene il valore da scrivere.<br>C#: nella struttura gestita, **Value** è costituito da un oggetto di tipo System::Object. Il tipo di oggetto deve essere coerente con le caratteristiche di **ValueRank** e **ValueType** dell'item.<br>Nel caso di scrittura di un solo elemento di un array è possibile specificare anche un valore singolo.<br><br>Es:<br>Valore singolo, Boolean: bool<br>Valore array, Boolean: bool[]<br>Valore singolo, Byte: byte<br>Valore array, Byte: byte[]<br>Valore singolo, SByte: sbyte<br>Valore array, SByte: sbyte[]<br>Valore singolo, UInt16: ushort<br>Valore array, UInt16: ushort[]<br>Valore singolo, Int16: short<br>Valore array, Int16: short[]<br>Valore singolo, UInt32: uint<br>Valore array, UInt32: uint[]<br>Valore singolo, Int32: int |

| | |
|---|---|
| Valore array, Int32: | int[] |
| Valore singolo, Float: | float |
| Valore array, Float: | float[] |
| Valore singolo, Double: | double |
| Valore array, Double: | double[] |
| Valore singolo, Bytestring: | byte[] |
| Valore array, Bytestring: | byte[][] |
| Valore singolo, String: | string |
| Valore array, String: | string[] |
| Valore singolo, LocalizedText: | string |
| Valore array, LocalizedText: | string[] |

C/C++: nella struttura non gestita, **Value** è un generico puntatore void* all'area di memoria che va predisposta, come il campo **ValueSize**, seguendo le indicazioni descritte nella funzione *OpcuacGetValue.*

**Valore di ritorno**

L'esito della scrittura; 0 (zero) se l'operazione è andata a buon fine altrimenti un codice di errore.
Per l'elenco dei possibili codici di errore fare riferimento al paragrafo "Errori della comunicazione OPC" a pag. 51.

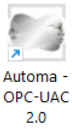
| | |
|---|---|
| Stato | Ok |
| Prodotto | OPC-UAC |
| Categoria | Funzioni di gestione valori pag. 23 |
| Referenze | A pag. 23 |
| Esempi | A pag. 23 |
| In OPC-UAC | Dalla versione 1.0 |

# 10 Applicativi demo

Sono forniti quattro progetti dimostrativi con funzionalità simili; i primi tre sono sviluppati in C++ e C# con Microsoft Visual Studio 2022 e 2010 mentre il quarto è realizzato con Python:

| | |
|---|---|
| OPCUACDemo_VC2022\OPCUACDemoCS | Applicazione C# Windows Form che utilizza l'interfaccia tipo assembly. |
| OPCUACDemo_VC2010\OPCUACDemoCpp | Applicazione C++ MFC dialog-based che utilizza l'interfaccia C/C++. |
| OPCUACDemo_VC2010\OPCUACDemoCS | Applicazione C# Windows Form che utilizza l'interfaccia C/C++ tramite P/Invoke.<br><br>Nel demo è presente il modulo OpcuacInterface.cs in cui è a disposizione il contenitore OpcuacInterface con la definizione di costanti, strutture e metodi di supporto all'utilizzo della DLL. |
| OPCUACDemo_Python312 | Applicazione realizzata con Python v. 3.12 |

Dall'elemento creato sul desktop:




è possibile accedere alla cartella selezionando il link "Applicativi dimostrativi".

Oppure è possibile accedere direttamente alla cartella inserendo nella finestra di "Esplora risorse" di Windows il seguente path:

%APPDATA%\Automa\Communication Tools\OPC-UAC 2.0\Demo

Il path corrispondente ad %APPDATA% dipende dal Sistema Operativo in uso.

Oltre alle cartelle dei demo è presente anche la cartella di servizio OPCUACFG:

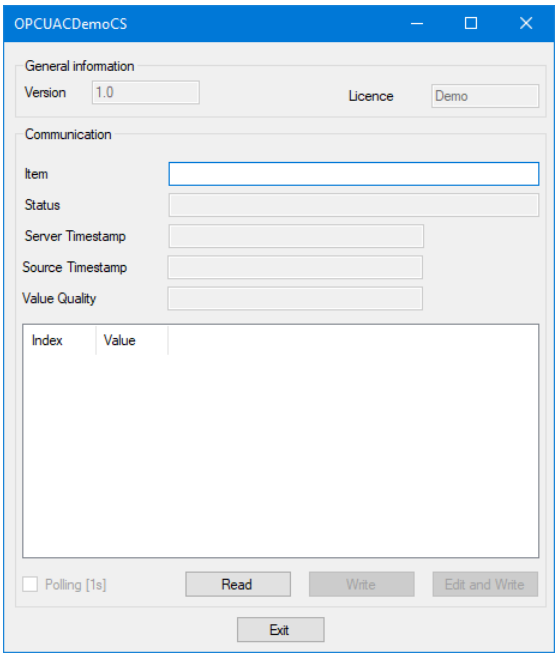
| | |
|---|---|
| OPCUACCFG | Cartella contenente tutti i file necessari per l'utilizzo del tool configuratore. |

## 10.1 Demo Microsoft Visual Studio

Tutti i progetti sono predisposti con le due configurazioni "Debug" e "Release".
Nelle rispettive cartelle target, impostate nei progetti anche come Working Directory, sono già presenti il file eseguibile e gli altri file necessari.
Per una primissima prova, anche fuori dall'ambiente di sviluppo, si consiglia di fare riferimento alla build "Release" del demo "OPCUACDemo_VC2022\OPCUACDemoCS\bin\Release".

Di default il file di configurazione "OPCUAC.XML" è vuoto.

Utilizzare il programma di configurazione, disponibile nella cartella OPCUACCFG e descritto nel paragrafo "Configurazione della basedati" a pagina 8, per predisporre una configurazione minimale (anche con un solo item); copiare il file "OPCUAC.XML" nella cartella di lavoro del demo e avviare l'applicazione.




Nella parte superiore sono visualizzate delle informazioni generali:

- Versione della libreria
- Stato della licenza: "Demo" se l'attivazione non è stata effettuata oppure "Ok". Nella modalità demo ad ogni lancio l'applicazione gestisce la comunicazione per un massimo di 15 minuti.

Inserire nel campo "Item" l'identificativo di uno degli item definiti nella basedati (per confermare il valore premere il tasto TAB) ; se la comunicazione con il server è stata stabilita e la subscription è stata correttamente attivata, nei campi sottostanti si visualizzeranno le informazioni relative al valore corrente dell'item.

I comandi disponibili sono:

- Opzione "Read": legge dal server il valore corrente dell'item ed aggiorna i dati a video.
- Opzione "Write": recupera dalla cache il valore corrente dell'item e lo riscrive interamente; nel campo "Status" viene visualizzato l'esito dell'operazione.
- Opzione "Edit and Write": gestisce l'input di uno dei valori (nel caso di array è necessario selezionare l'elemento che si intende modificare) ed effettua la scrittura dell'item; nel campo "Status" viene visualizzato l'esito dell'operazione.
- Opzione "Polling": ogni secondo recupera dalla cache il valore corrente dell'item.

**Note**
Per la modifica dell'applicazione OPCUACDemo_VC2022\OPCUACDemoCS è necessario avere installato sul PC .Net Framework 4.6.2 Developer pack.

## 10.2 Demo Python

Di default il file di configurazione "OPCUAC.XML" è vuoto.

Utilizzare il programma di configurazione, disponibile nella cartella OPCUACCFG e descritto nel paragrafo "Configurazione della basedati" a pagina 8, per predisporre una configurazione minimale (anche con un solo item); copiare il file "OPCUAC.XML" nella cartella di lavoro del demo e avviare l'applicazione.


Nella parte superiore sono visualizzate delle informazioni generali:

- Versione della libreria
- Stato della licenza: "Demo" se l'attivazione non è stata effettuata oppure "Ok". Nella modalità demo ad ogni lancio l'applicazione gestisce la comunicazione per un massimo di 15 minuti.

Inserire nel campo "Item name" l'identificativo di uno degli item definiti nella basedati e premere il pulsante "Read": se la comunicazione con il server è stata stabilita e la subscription è stata correttamente attivata, nei campi a destra si visualizzeranno le informazioni relative al valore corrente dell'item.

Per scrivere il valore, editare un valore coerente con il tipo di Item nel campo "Write value" e premere il pulsante "Write": l'esito dell'operazione sarà visualizzato nel campo "Status".

**Note**

Per il funzionamento di questo esempio è necessario aver installato sul PC una versione di Python a **32 bit** e la libreria "pythonnet".

# 11 Appendice

## 11.1 Status del valore

I possibili valori dello status associato al valore del dato sono elencati nella tabella che segue. Tramite la funzione *OpcuacValueStatusInfo* è possibile ottenere l'indicatore di usabilità associato a un determinato codice (Good/Uncertain/Bad).

| Value Status | Codice Dec/Hex | Descrizione breve | Descrizione |
|---|---|---|---|
| BAD | 2147483648<br>80000000H | Bad | |
| | 2156462080<br>80890000H | BadConfigurationError | There is a problem with the configuration that affects the usefulness of the value. |
| | 2156527616<br>808A0000H | BadNotConnected | The variable should receive its value from another variable; but has never been configured to do so. |
| | 2150694912<br>80310000H | BadNoCommunication | Communication with the data source is defined; but not established; and there is no last known value available. |
| | 2156724224<br>808D0000H | BadOutOfService | The source of the data is not operational. |
| | 2156593152<br>808B0000H | BadDeviceFailure | There has been a failure in the device/data source that generates the value that has affected the value. |
| | 2156658688<br>808C0000H | BadSensorFailure | There has been a failure in the sensor from which the value is derived by the device/data source. |
| | 2150760448<br>80320000H | BadWaitingForInitialData | Waiting for the server to obtain values from the underlying data source. |
| | 2156789760<br>808E0000H | BadDeadbandFilterInvalid | The deadband filter is not valid. |
| UNCERTAIN | 1073741824<br>40000000H | Uncertain | |
| | 1083113472<br>408F0000H | UncertainNoCommunicationLastUsableValue | Communication to the data source has failed. The variable value is the last value that had a good quality. |
| | 1083179008<br>40900000H | UncertainLastUsableValue | Whatever was updating this value has stopped doing so. |
| | 1083244544<br>40910000H | UncertainSubstituteValue | The value is an operational value that was manually overwritten. |
| | 1083310080<br>40920000H | UncertainInitialValue | The value is an initial value for a variable that normally receives its value from another variable. |
| | 1083375616<br>40930000H | UncertainSensorNotAccurate | The value is at one of the sensor limits. |

| Value Status | Codice Dec/Hex | Descrizione breve | Descrizione |
|---|---|---|---|
| | 1083441152 40940000H | UncertainEngineeringUnitsExceeded | The value is outside of the range of values defined for this parameter. |
| | 1083506688 40950000H | UncertainSubNormal | The value is derived from multiple sources and has less than the required number of Good sources. |
| GOOD | 9830400 00960000H | GoodLocalOverride | The value has been overridden. |

## 11.2 Status dell'esecuzione di un metodo

I codici principali per l'esito sono elencati nella tabella che segue nella colonna "Codice".
Eventuali altri codici negativi non presenti in tabella devono essere considerati come esito "BAD".

| Esito | Codice Dec/Hex | Descrizione breve | Descrizione |
|---|---|---|---|
| BAD | -2136276992 80AB0000H | BadInvalidArgument | At least one input argument broke a constraint (e.g. wrong data type, value out of range) |
| | -2145452032 801F0000H | BadUserAccessDenied | User does not have permission to perform the requested operation. |
| | -2143617024 803B0000H | BadNotWritable | The access level does not allow writing to the Node. |
| | -2143289344 80400000H | BadNotImplemented | Requested operation is not implemented. |
| | -2139881472 80740000H | BadTypeMismatch | The value supplied for the attribute is not of the same type as the attribute's value. |
| | -2139750400 80760000H | BadArgumentsMissing | The client did not specify all of the input arguments for the method. |
| GOOD | 000000000 00000000H | Good | |

## 11.3 Codici di errore

I codici di errore si dividono nelle seguenti categorie:

- Errori generali (inizializzazione)
- Errori della comunicazione OPC UA

### 11.3.1 Errori generali

Questi sono gli errori che possono essere segnalati in fase di inizializzazione del driver.

| Errore Dec | Errore Hex | Descrizione |
|---|---|---|
| -0001 | FFFF | Errore interno di gestione memoria.<br><br>Possibili cause/interventi:<br>• La libreria non è stata correttamente installata nell'applicativo: vedere "Deployment" a pag. 4. |
| -0110 | FF92 | Fine funzionamento dimostrativo.<br><br>Possibili cause/interventi:<br>• La licenza della libreria non è attiva; fare riferimento al paragrafo "NDP462-KB3151800-x86-x64-AllOS-ENU.exe<br>• VC_redist.x86.exe<br><br>Il "Certificate Store", se non predisposto preventivamente, viene creato automaticamente alla prima esecuzione del tool di configurazione o dell'applicazione.<br>• Attivazione licenza della libreria" a pag. 5. |
| 0001 | 0001 | La libreria non è stata inizializzata.<br><br>Possibili cause/interventi:<br>• Assicurarsi di aver chiamato la funzione *OpcuacInit*. |
| 4097 | 1001 | Il file OPCUAC.Config.XML è mancante.<br><br>Possibili cause/interventi:<br>• La libreria non è stata correttamente installata nell'applicativo. Ripristinare il file dalla cartella "DRIVER" del path di installazione: vedere "Deployment" a pag. 4. |
| 4098 | 1002 | Il file OPCUAC.Config.XML non è valido.<br><br>Possibili cause/interventi:<br>• La libreria non è stata correttamente installata nell'applicativo. Ripristinare il file dalla cartella "DRIVER" del path di installazione: vedere "Deployment" a pag. 4. |
| 4099 | 1003 | Il certificato dell'applicazione è mancante. |
| 4100 | 1004 | Il file di configurazione della basedati OPCUAC.XML è mancante. |
| 4101 | 1005 | Il file di configurazione della basedati OPCUAC.XML non è valido. |

### 11.3.2 Errori della comunicazione OPC UA

Questi errori possono essere relativi a:
- problemi nella comunicazione con il server
- problemi relativi ai parametri nelle richieste di lettura e scrittura degli item

| Errore Dec | Errore Hex | Descrizione |
|---|---|---|
| 8193 | 2001 | Il collegamento con il server non è attivo.<br><br>Possibili cause/interventi:<br>• il server non è attivo<br>• URL di connessione non corretto<br>• aumentare il valore del parametro "Session Timeout"<br>• risolvere eventuali interferenze di firewall e/o antivirus<br>• utilizzare il tool configuratore e verificare eventuali segnalazione di errori nella barra di stato<br>• verificare che il server abbia accettato il certificato della libreria OPC-UAC |
| 8194 | 2002 | Il collegamento con il server non è attivo per un problema legato al certificato.<br><br>Possibili cause/interventi:<br>• il certificato del server non è più valido: attivare il tool configuratore OPCUACCFG.EXE, accedere all'opzione "Configuration" nelle proprietà del server interessato, eseguire l'operazione di "Refresh" e risalvare la configurazione. |
| 8195 | 2003 | L'item richiesto non esiste. |
| 8196 | 2004 | L'item richiesto non è attivo. |
| 8197 | 2005 | La lettura dell'item è fallita (ReadValue service non disponibile o in errore). |
| 8198 | 2006 | Il valore dell'item non è ancora disponibile. |
| 8199 | 2007 | La scrittura dell'item è fallita (WriteValue service non disponibile o in errore). |
| 8200 | 2008 | La scrittura dell'item è fallita (errore scrittura su device).<br>Una possibile causa è l'incoerenza tra il tipo di dato dichiarato nell'item e il tipo di dato passato alla funzione .NET. |
| 8201 | 2009 | La procedura di SafeWrite è fallita (il client non ha riletto il valore modificato entro il timeout di SafeWrite).<br><br>Possibili cause/interventi:<br>• aumentare il valore del parametro "Safe Write timeout" del server<br>• disabilitare l'opzione "Safe Write" nell'item |
| 8202 | 200A | Lettura/scrittura item: l'item non contiene un valore di tipo array.<br><br>Possibili cause/interventi:<br>• verificare i parametri **numval** e **index** (per i valori singoli index deve essere 0 e numval può essere 0 oppure 1) |
| 8205 | 200D | Lettura/scrittura item: il tipo di dato OPC UA dell'item non è supportato. |
| 8206 | 200E | Lettura/scrittura item: parametri non validi.<br><br>Possibili cause/interventi:<br>• verificare che i parametri di **item** e **pdBuf** non siano NULL. |

| Errore Dec | Errore Hex | Descrizione |
|---|---|---|
| 8208 | 2010 | Lettura/scrittura item: l'indice specificato per l'acceso all'array è fuori dal range.<br><br>Possibili cause/interventi:<br>• verificare il parametro **index**. |
| 8209 | 2011 | Scrittura item da interfaccia C/C++: la dimensione dichiarata per il buffer contenente i dati non è coerente rispetto al numero di dati specificati.<br><br>Possibili cause/interventi:<br>• verificare il parametro **numval** e il campo **ValueSize**. |
| 8210 | 2012 | Lettura/scrittura item: il numero di valori richiesti è maggiore di quelli contenuti nell'item.<br><br>Possibili cause/interventi:<br>• verificare il parametro **numval**. |
| 8211 | 2013 | Lettura/scrittura item: l'attuale valore dell'item non è valido (DataValue è NULL). |
| 8214 | 2016 | La chiamata al metodo è fallita.<br><br>• Per informazioni di dettaglio fare riferimento al codice di dettaglio dell'esito del metodo. |

# 12 Troubleshooting

| Configuratore: è visualizzato l'errore "Server not connected" |
|---|
| - verificare che l'indirizzo IP del server sia raggiungibile<br>- verificare l'URL di collegamento<br>- se nell'URL si sta utilizzando un nome DNS assicurarsi che il nome venga risolto con l'indirizzo IP del server (es. tramite dichiarazione nel file HOSTS)<br>- verificare che il server abbia accettato (trusted) il certificato OPCUACCFG del configuratore<br>- verificare eventuali limiti del server: es. numero massimo di subscriptions supportate, numero massimo di item nelle subscription<br>- alzare il tempo di "Session timeout" |

| Configuratore: è visualizzato l'errore "Server not connected – BadCertificateHostNameInvalid". |
|---|
| In base a quanto previsto per la gestione sicurezza, nel campo "Subject Alternative Name" del certificato del server dovrebbe essere presente l'indirizzo IP o un nome DNS del server stesso.<br><br>Per connettersi al server occorre poi utilizzare un URL coerente a tale informazione per cui ad esempio se nel certificato l'hostname è indicato come:<br>- indirizzo IP "192.168.0.100": l'URL da usare sarà tipo "opc.tcp://192.168.0.100:4840"<br>- nome DNS "ServerOPCUA": l'URL da usare sarà tipo "opc.tcp://ServerOPCUA:4840"<br><br>La procedura standard di sicurezza di OPC UA prevede che in fase di connessione il client esegua un controllo sulla coerenza tra l'URL e quanto presente nel certificato e in caso di non corrispondenza interrompa la connessione con un errore di "BadCertificateHostNameInvalid".<br><br>In questa situazione occorre:<br>- verificare l'URL che si sta usando<br>- verificare il contenuto del certificato del server<br><br>Se il problema è che nel certificato del server è dichiarato un errato indirizzo IP e ci si trova impossibilitati a far rigenerare un certificato corretto è possibile abilitare la flag "Disable host name check" per inibire il controllo. |

| Configuratore: è visualizzato l'errore "Server not connected - check certificate - BadCertificateUntrusted". |
|---|
| Come default il programma configuratore e il driver runtime sono configurati per accettare automaticamente i certificati self-signed; di massima quindi non dovrebbero presentarsi problemi di connessione legati alla mancata accettazione dei certificati dei server.<br>Nel caso di problemi:<br>- accedere all'opzione "Configuration" nelle proprietà del server interessato, eseguire l'operazione di "Refresh" e risalvare la configurazione<br>- verificare la presenza di eventuali certificati presenti nella cartella "Rejected" del Certificate Store (cartella Automa\pki\rejected) e spostarli eventualmente nella cartella "Trusted" (l'operazione va fatta mentre il software non è attivo) |

| Runtime: presenza di errori |
|---|
| - consultare l'elenco degli errori nella sezione "Codici di errore" a pag. 50.<br>- nel caso di problemi relativi alla connessione provare ad utilizzare il tool configuratore per indagare ulteriormente il problema |

# 13 Note dell’utente