Modalità RDS e modernizzazione delle architetture InTouch stand alone

Rev. 2 – 28/08/2026

Runtime “Master” e “Slave”

Partendo da un’architettura di supervisione composta da diverse postazioni fisse dedicate (PC), ognuna delle quali esegue localmente una copia della stessa applicazione InTouch, consideriamo lo scenario in cui un PC è eletto a nodo principale (“master”) e gli altri PC svolgono il ruolo di “slave”; usualmente, il nodo master è il PC che ospita i driver di comunicazione verso il campo, ed è l’unico a svolgere alcune funzioni, ad esempio la centralizzazione del riconoscimento degli allarmi.
Ciascun PC dell’architettura acquisisce la propria licenza di tipo *InTouch Workstation*, per esporre all’operatore una singola interfaccia runtime WindowViewer che carica il medesimo progetto.
Pertanto, tale progetto InTouch dovrà includere un check iniziale che inibisca l’esecuzione delle funzionalità master ogni volta che l’applicazione viene avviata su un nodo slave; questa distinzione può essere ad esempio basata su *hostname*, o su qualche altro identificatore equivalente.

L’architettura fin qui descritta può essere modernizzata traendo vantaggio dalla tecnologia Remote Desktop Services (RDS), già nota come TSE (Terminal Server Edition), messa a disposizione dai sistemi operativi Windows Server; in tale approccio, le sessioni WindowViewer sono eseguite su una macchina server (fisica o virtuale), ma vengono utilizzate da nodi Thin Client in rete mediante connessione Remote Desktop Protocol (RDP).
Questo genere di transizione, o concezione di nuovi progetti, viene in particolare incentivato dall’introduzione della tipologia di licenza *InTouch Unlimited*, grazie alla quale non è più necessario acquistare una licenza Workstation per ogni sessione WindowViewer, in quanto è già incluso un numero virtualmente illimitato di sessioni client InTouch RDS (il limite reale sul numero di client contemporanei dipende principalmente dalle risorse hardware del terminal server).

Nella modalità RDS, si avranno comunque le sessioni WindowViewer master e slave. Solitamente, l’applicazione master viene eseguita in *sessione console* di Windows, mentre i runtime slave vengono fruiti tramite sessioni RDP; in questo caso, il metodo per distinguere l’avvio in modalità master o slave preferibilmente non è basato sul check dell’hostname, ma piuttosto sulla discriminazione, appunto, tra esecuzione in sessione console o RDS. Per fare questo, si può usare una delle seguenti funzioni all’interno dello script di startup dell’applicazione:

TseQueryRunningOnConsole()

TseQueryRunningOnClient()

La prima query restituisce valore 1 (*true*) quando ci si trova in sessione console, e 0 (*false*) altrimenti, esattamente il contrario di ciò che viene restituito dalla seconda query.
Sono inoltre abilitate in ambiente RDS le seguenti funzioni script:

TseGetClientId()

TseGetClientNodeName()

Per approfondimenti, consultare la sezione "*Retrieve information about the InTouch client session using scripts*" della documentazione ufficiale.

Oltre all'adattamento del check per la distinzione del ruolo master/slave, poiché tutte le sessioni WindowViewer vengono di fatto eseguite sulla stessa macchina, in modalità RDS dovranno essere previste alcune accortezze aggiuntive, onde evitare conflitti tra processi e malfunzionamenti, come viene esposto nei paragrafi a seguire.

Storicizzazione dei Dati

Valutando lo scenario tipico in cui sia stata configurata nel progetto InTouch la storicizzazione in formato file .LGH all'interno di una directory specifica (es. "C:\Trend"), si evince facilmente che in modalità RDS la funzione di storico dovrà essere demandata unicamente all'istanza *master* ed inibita nelle sessioni *slave*, altrimenti più processi accederebbero contemporaneamente in scrittura allo stesso file nella stessa cartella, corrompendolo nel tentativo di registrare dati ridondanti.
All'avvio in modalità slave, quindi, l'applicazione dovrà immediatamente *azzerare* la tag di sistema **$HistoricalLogging** (vedi *"Start and stop historical logging at run time"* in documentazione).

Tuttavia, ciò non è sufficiente, poiché sussiste sempre un minimo ritardo tra il lancio del runtime WindowViewer e l'esecuzione dello script di startup dell'applicazione, quindi esiste un brevissimo lasso di tempo all'interno del quale la tag $HistoricalLogging non è stata ancora azzerata, e l'applicazione slave può accedere per un attimo in scrittura al file LGH. Per evitare questo, è necessario assicurarsi che sia presente la seguente riga nella sezione *[InTouch]* del file **InTouch.ini** all'interno della cartella del progetto:

```
SkipInitialHistdata=1
```

Nota: questi stessi concetti si applicano anche laddove sia configurata la storicizzazione diretta dei dati da progetto InTouch a server AVEVA **Historian**; in tal caso, pur non essendovi il pericolo di corruzione dei file relativi agli History Block, *senza le suddette accortezze non si potrebbe evitare la problematica della registrazione di dati ridondanti/duplicati*.

Storicizzazione e Gestione degli Allarmi

Considerando l'archiviazione di allarmi (ed eventi) in istanza MS SQL Server (database usualmente denominato "**WWALMDB**"), la storicizzazione in questo caso non viene eseguita direttamente dai processi WindowViewer (*view.exe*), bensì dal processo Alarm DB Logger (***wwalmlogger***.*exe*) che fa da "intermediario"; questa funzionalità viene configurata tramite wizard *Alarm DB Logger Manager* (almlogwiz.exe), che per default propone la Alarm Query "*\InTouch!$System*"; tale query, se lasciata inalterata, fa sì che il processo wwalmlogger si agganci all'Alarm Provider corrispondente all'istanza view.exe *della stessa sessione di Windows che avvia wwalmlogger*, per scaricare le entry di allarme relative al gruppo predefinito $System e a tutti i suoi eventuali sottogruppi definiti in modo custom.

Nota: WindowViewer può essere lanciato prima o dopo wwalmlogger, può essere chiuso e riaperto più volte, e viceversa; fintanto che entrambi i processi View e wwalmlogger sono in esecuzione contemporaneamente, la storicizzazione avverrà; inoltre, le entry di allarme vengono bufferizzate e possono essere recuperate in un secondo momento, nel caso wwalmlogger non sia attivo mentre View gestisce un qualsiasi insieme di cambi di stato degli allarmi.
Oltretutto, al momento del lancio, il processo wwalmlogger esegue di per sé stesso un check, che ne causa l'immediato arresto nel caso sia già in esecuzione un'altra istanza dello stesso processo in una qualsiasi sessione sulla macchina.

Nella trattazione finora esposta, l'istanza WindowViewer in sessione *console* coincide con il runtime *master*; restando in tale ambito, consideriamo come tipico lo scenario in cui anche wwalmlogger venga avviato nella sessione console, con query di default, in modo che si colleghi alla sessione View master. Dalle varie considerazioni di cui sopra segue che, nella configurazione più comune, non è possibile avviare in alcun modo più di un processo wwalmlogger, e l'unico processo wwalmlogger preleva allarmi dalla sola istanza InTouch master.
Normalmente, dunque, non vi è pericolo di archiviare allarmi/eventi duplicati in WWALMDB. Per evitare invece perdite di dati/allarmi storici, la cosa di cui bisognerà assicurarsi è semplicemente che sia l'istanza View master, sia il processo wwalmlogger si avviino automaticamente in sessione console e restino in esecuzione il più costantemente possibile.

Diversamente da quanto suddetto, se nel progetto InTouch è stata abilitata l'opzione:
Historical logging > Historian logging > Enable storage to historian
con spunta "Log alarms and events" attiva, allora potranno essere registrati allarmi duplicati negli History Block, in quanto ogni processo WindowViewer è in grado di inviare autonomamente dati al server Historian. Onde evitare record duplicati, poiché azzerare *$HistoricalLogging* non ha effetto nel caso della storicizzazione allarmi, è necessario impedire totalmente il processamento di questi ultimi *nelle applicazioni con ruolo slave*, *azzerando* in esse la proprietà **.AlarmEnabled** del gruppo radice *$System*, per inibire la rilevazione degli allarmi appartenenti a tale gruppo e a tutti i sottogruppi; gli allarmi (e gli eventi) verranno quindi convenientemente processati solo a livello dell'applicazione *master*, mentre la visualizzazione degli allarmi attivi resterà funzionante anche nelle istanze *slave*, fintanto che l'Alarm Client avrà una query del tipo "**\\:**\InTouch!<*group*>", così da puntare sempre al *runtime master come provider*, garantendo al contempo la *gestione centralizzata degli acknowledge*, ove richiesto dal progetto. La parte fondamentale è la presenza del carattere "due punti" dopo il doppio backslash iniziale, per rendere possibile l'accesso specificamente al provider della sessione console; la sintassi della query, infatti, è "**\\<*host*>:<*session*>**\InTouch!<*group*>", dove <host> identifica la macchina e se lasciato *vuoto* equivale a scrivere *localhost*, mentre <session> è l'indirizzo IP del client RDP *oppure una stringa vuota per indicare la sessione console*. Se non si inserisse il carattere "due punti", scrivendo ad esempio "\\localhost\InTouch!$System", ogni client si aggancerebbe comunque al runtime InTouch della propria sessione, ovvero a sé stesso, di fatto avendo un comportamento del tutto equivalente alla query "\InTouch!$System".

Nota: l'unica tipologia di autenticazione funzionante da impostare nell'Alarm Client in modalità storico per l'accesso agli eventi archiviati negli History Block è la "**Windows integrated**"; le altre modalità possono essere selezionate, ma causano il mancato caricamento della lista a runtime.

Per quanto riguarda invece gli allarmi archiviati in SQL Server (WWALMDB), l'eventuale impostazione di "Authentication Mode: *Windows Account*" *viene di fatto ignorata* sia a design time che a run time, e il sistema si comporta come fosse selezionata comunque la "Windows integrated".
Per questi motivi, considerando i problemi (recentemente anche di sicurezza) dell'autenticazione "SQL Server", *si raccomanda di scegliere sempre la modalità "Windows integrated", configurando opportunamente i permessi d'accesso dei vari utenti di Windows* nell'istanza SQL Server (ad esempio tramite SQL Server Management Studio), sia nel caso History Block che WWALMDB.

Esempi e Raccomandazioni Conclusive

Di seguito viene presentata una porzione di codice d'esempio che può essere utilizzata all'interno dello script *Application - Condition Type > On Startup* e che combina assieme vari concetti rilevanti presentati nei paragrafi precedenti:

```
if TseQueryRunningOnClient() then

    {Slave: disattivo la storicizzazione dati}
    $HistoricalLogging = 0;

    {Slave: disattivo la rilevazione degli allarmi}
    $System.AlarmEnabled = 0;

endif;
```

Nota: onde evitare accessi concorrenti alla directory dell'applicazione, si raccomanda di *abilitare sempre la funzionalità "Network application development" (NAD) per ogni utente* del sistema - vedi "*Configure an InTouch application for NAD*" in documentazione.

Si ricorda, infine, che le licenze per l'abilitazione delle sessioni RDS di Windows (User CALs - Client Access Licenses) vanno acquistate separatamente, attraverso i canali commerciali Microsoft.
La tipologia di licenza Per-Device (Device CALs) *non* è generalmente supportata.

Versioni e Riferimenti

La presente trattazione è valida per la versione InTouch **2023 R2 SP1 P04**.

Sono di particolare rilievo le seguenti aree della documentazione di InTouch:

"Deploy and work with Terminal Services and Remote Desktop Services"

InTouch HMI 2023 R2 SP1 release notes > ***Resolved issue 3336226***

InTouch HMI 2023 R2 SP1 P04 release notes > ***Resolved issue 4296775***