# Guardian: Descrizione tecnica

***Release: 1.4 – 24 giugno 2010***

***Autore: Andrea Gualducci***

***Modifiche***

*1.0 → 1.1*

- Aggiunta capitolo Topologia

*1.1 → 1.2*

- Aggiunta descrizione popup-video sul client

*1.2 → 1.3*

- Aggiunti Report Parametrici
- Aggiunto backup ContactID

*1.3 → 1.4*

- Estensione Building automation

MetaRec SRL
security integration

## SOMMARIO

## Introduzione

Il supervisore Guardian rappresenta l'ultima evoluzione dei supervisori per ***building security*** proposti e sviluppati da MetaRec s.r.l.

### Portabilità

Esso si basa sulla architettura JaMIX™, un insieme di moduli scritti in Java che possono essere eseguiti su qualunque sistema operativo (Windows, Linux, Unix, Macintosh) e possono essere configurati per svolgere qualunque compito di supervisione.

### Modularità

I moduli che compongono Guardian possono essere eseguiti tutti su una singola macchina, come su macchine differenti a seconda delle esigenze, permettendo così di calibrare e distribuire nel miglior modo possibile il carico computazionale che le macchine devono sopportare.

### Java

Questo supervisore trae vantaggio dal linguaggio Java in cui è interamente scritto il quale permette di ottenere codice molto più sicuro e pulito, con minore probabilità di errori di programmazione e maggiore facilità di gestione dei progetti.

### Grafica avanzata

Guardian permette di gestire anche una interfaccia grafica animata che visualizza le mappe degli ambienti controllati e da queste è possibile gestire integralmente tutti i sensori antiallagamento, antincendio e antintrusione dislocati.
Guardian permette l'utilizzo delle mappe come interfaccia di controllo fondamentale: navigando tra le mappe, è possibile tenere sotto monitoraggio visivo tutti i sensori presenti negli edifici controllati e agire su di essi per riconoscere gli allarmi, ripristinarli, o per compiere verifiche.
Guardian inoltre gestisce in modo più pulito le azioni da compiere sulle centrali: queste sono raggruppate in uno speciale sinottico grafico che mostra con simboli di facile ed intuitiva interpretazione problemi di disconnessione, particolari stati di verifica e controllo in cui esse possono trovarsi.

### Diagnostica interna

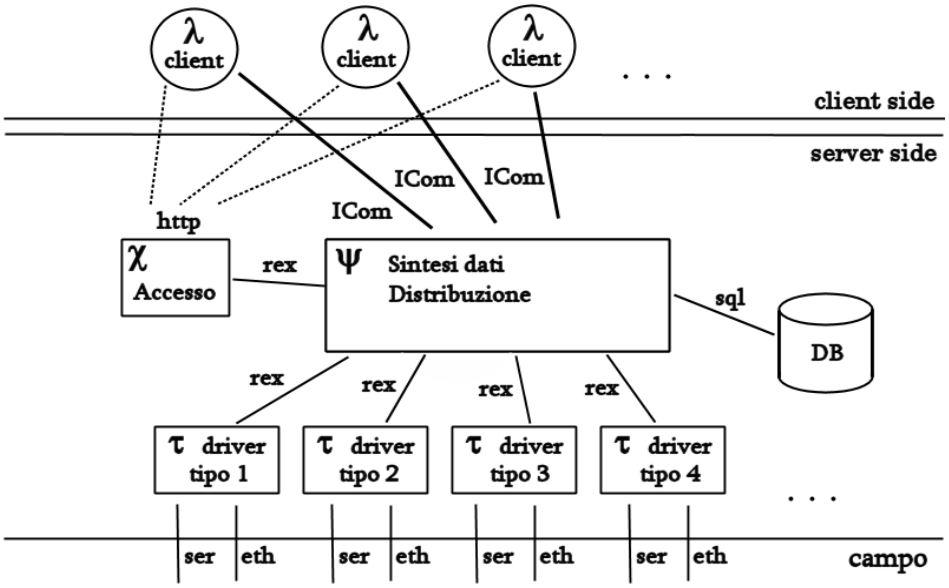
Guardian è inoltre dotato di un particolare sistema software che gli permette di tenere sotto controllo tutti i moduli di cui è composto: se uno di questi moduli per qualunque motivo smette di funzionare o si trova in uno stato non risolvibile, il sistema provvede a rimpiazzarlo e in pochi secondi viene ripristinata la funzionalità originale.

### Integrazione video

Associando i sensori alle telecamere dislocate sull'impianto, Guardian permette di corredare gli eventi di allarme con immagini contestuali, attraverso un sistema di popup. Il video dell'evento può essere visualizzato in modo automatico o richiamato dall'operatore su eventi presenti, eventi storici o dalle mappe.

## Architettura del server

Il server di Guardian si compone di diversi moduli, ciascuno dei quali costituisce un processo separato del sistema operativo. Poiché tutti i moduli sono applicativi java, tutti questi processi appaiono come macchine virtuali java (JVM).
Nella seguente immagine viene rappresentato uno schema a blocchi dei moduli




### PSI – Sintesi dati e Distribuzione

Il modulo fondamentale che ospita tutti le unità di sintesi dei dati. Fondamentalmente si tratta di un RTDB (Real-Time Data Base), ma con una struttura di scambio dati basata sul push delle variazioni, caratteristica che lo rende notevolmente efficiente e capace di pesanti elaborazioni con minimo consumo di CPU.
Le unità di sintesi dei dati sono delle singole classi java che implementano sostanzialmente una tabella relazionale e possono essere programmate per svolgere qualunque compito ed interazione. Il linker dinamico di java fornisce l'ambiente ideale per questo tipo di approccio.
Questo modulo si connette a tutti gli altri moduli interni tramite il protocollo ReX (Record eXchange), ai moduli client tramite il protocollo InterCOM (abb. ICom) e ad un database ralazionale tramite JDBC.
Per dettagli sui protocolli interni, si rimanda agli allegati.

### CHI – Server di accesso

Fondamentalmente si tratta di un WebServer ed ha il compito fondamentale di far partire e tenere sotto controllo tutti gli altri moduli del server e di fornire la diagnostica interna.
Verso i client si comporta come un server di accesso, validando le credenziali dell'utente e rimandando il collegamento dati al server di distribuzione più appropriato.
Tramite questo server viene gestita l'anagrafica utenti del sistema.

### TAU – I driver

Per ogni tipologia (protocollo) di dispositivo di campo configurato su Guardian, viene fatto partire un driver che gestirà tutti i dispositivi di quel tipo, fornendo una interfaccia fra la sintesi dei dati ed il campo.

Tutti i driver di Guardian possono aprire canali seriali (COM) o ethernet (TCP o UDP) di comunicazione verso i dispositivi controllati, che si possono quindi remotizzare compatibilmente con le impostazioni di routing della rete in uso.
Viene fornita di seguito una lista dei driver a corredo di Guardian, relativa alla data del presente documento.

**Notifier AM2020**

Vecchia centrale antincendio di Notifier, da tempo non più prodotta e anche la sua supervisione è in disuso (sostituita generalmente con AM6000). Questo è l'unico driver JNI di Guardian e ha una limitazione strutturale di una centrale per driver. La supervisione è comunque completa. La centrale può dare problemi su riconnessione che rendono necessario il riavvio della stessa.

**Notifier AM6000-AM2000**

Centrale antincendio estremamente pratica, con un ottimo protocollo di interfacciamento che è una semplificazione del CEI-ABI. Supervisione completa, ma priva di gestione anagrafiche, effettuata su SIB600 (richiesta). Non è prevista una gestione delle aree per centrali antincendio

**Boselli CP100**

Centrale antintrusione. Antica centrale, una delle prime con protocollo APLEX. E' l'antenato della Hesa-Metro. Funzionalità identiche al driver per Hesa-Metro. Nessuna gestione aree. Gestione anagrafica molto limitata: anagrafica codici solo in lettura.

**Hesa Metro**

Centrale antintrusione. Di fatti lo stesso driver della Boselli CP100. La centrale è fisicamente migliore e la supervisione è migliore, ma le funzionalità sono identiche.

**Hesa Primato e Primato30**

Centrale antintrusione. Sempre protocollo APLEX, ma con estensione a 200 punti invece che 100. Può essere utilizzata in modalità con o senza aree. Gestione anagrafica molto più avanzata rispetto al driver Hesa-Metro. Il driver inoltre accetta anche la versione Primato 30, ultima evoluzione del firmware da Hesa. La supervisione è ottimale per questa centrale.

**Hesa SigNET**

Centrale antintrusione. Dispositivo completamente rivisto e riprogettato rispetto alle Metro e Primato.
La supervisione di questa centrale viene fatta direttamente tramite ethernet sulla centrale, senza il bisogno di passare attraverso un serial-hub, caratteristica che rende enormemente più veloci tutte le operazioni. Tuttavia il protocollo di trasporto è UDP, non TCP, quindi particolare attenzione al routing.
Alla data del presente documento le carenze del protocollo obbligano ancora ad una definizione manuale delle associazioni area-sensore.

**Bosch BZ500**

Centrale antintrusione. Protocollo UAZ2000 di Bosch. Ha una gestione degli stati molto particolare, tanto che tutti i sensori vengono esposti sia come sensori di allarme che come tecnologici dal driver.
La gestione anagrafiche si limita alla lista punti.

**Bosch DIVAR**

Server video. Driver di diagnostica per video server DIVAR. Non riporta il video, ma tiene solo sotto controllo il videoserver comunicando situazioni limite, come perdita telecamera, riempimento disco, ecc.

**GE-Interlogix CD150**

Centrale antintrusione. Supervisione effettuabile tramite modulo apposito di Aritech MPI232 (fuori produzione) oppure tramite modulo della SIDES CDP10. La supervisione tramite modulo SIDES è completa, fornisce tutte le anagrafiche ed è ottimale.
La gestione anagrafiche comprende la lista punti, la lista aree, la lista utenti.
Alto livello di affidabilità.

**GE-Interlogix Master**

Centrale antintrusione. Protocollo molto avanzato, gestione centrale molto semplice ed efficace. Il modulo di comunicazione permette una gestione in multidrop su RS485.

**GE-Interlogix FP2000**

Centrale antincendio. Protocollo molto avanzato, gestione centrale completa, con 8 tipi di anagrafica. La centrale può essere gestita direttamente o in relay collegandosi ai moduli concentratori, caso in cui può essere interfacciata l'intera rete ARCNET con una sola seriale.

**GE-Interlogix CSx75**

Centrale antintrusione di piccolo-medio taglio e dal costo contenuto.

**Honeywell Galaxy**

Centrale antintrusione. Per nuovo e vecchio firmware. Buona supervisione con gestione anagrafiche. Tutte le notifiche del protocollo SIA devono essere attivate sulla centrale e il livello deve essere posto a 3.
Le centrali gestite sono sia le classic (60, 504 et al.), sia le G2 e G3 che le nuove Dimension per le quali sono implementati anche la storicizzazione degli accessi e gli allarmi per accessi rifiutati.

**CIAS SA2ISI**

Centrale antintrusione con protocollo R3, tramite scheda SA3COM. La limitazione dell'allineamento fornita dalla centrale rende tale supervisione poco precisa e da evitare dove possibile. Attenzione sia alla scheda SA3COM (deve essere la versione giusta) e ad alcuni moduli interni prodotti da CIAS senza compatibilità.

**Raychem TTDM**

Centrale antiallagamento. Nelle versioni TTDM, TTDM Plus.
La gestione dell'evento associa un punto di allarme per ogni range (configurabile) sul cavo sensore. Nel caso della TTDM+ è necessario configurare la centrale in lingua inglese e lasciare il pannello nella schermata principale.

**Advantech Adam5000**

Modulo di IO di rete. Permette di associare la gestione degli IO (input e relé) a situazioni complesse di allarme/inserimento aree/sensori.

**Advantech PCI1762**

Modulo di IO scheda interna. Permette di associare la gestione degli IO (input e relé) a situazioni complesse di allarme/inserimento aree/sensori.

**Trend IQ3**

Modulo di regolazione meccanica (building automation) di Trend (gruppo Honeywell). Interfacciamento XML agli elementi sWitch, Knob, digital Input, Analog input, Sensor, Driver. Permette di scrivere valori su K e W.

**STS-Elettronica FD68**

Centrali di piccolo taglio, antintrusione e antincendio. Gestione completa. Possibilità di interfacciamento diretto, oppure off-line tramite modem con chiamate in uscita o in entrata da centrale.

**ContactID**

Per Ademco 685 o altri ricevitori ContactID, non è un driver autonomo: recupera informazioni che gli altri driver useranno come backup o come integrazione dati.

**MC55**

Driver di gestione modem con protocollo AT esteso (MC55) utilizzato per l'invio di SMS, collegabili agli allarmi, tramite apposito sottosistema di Guardian.

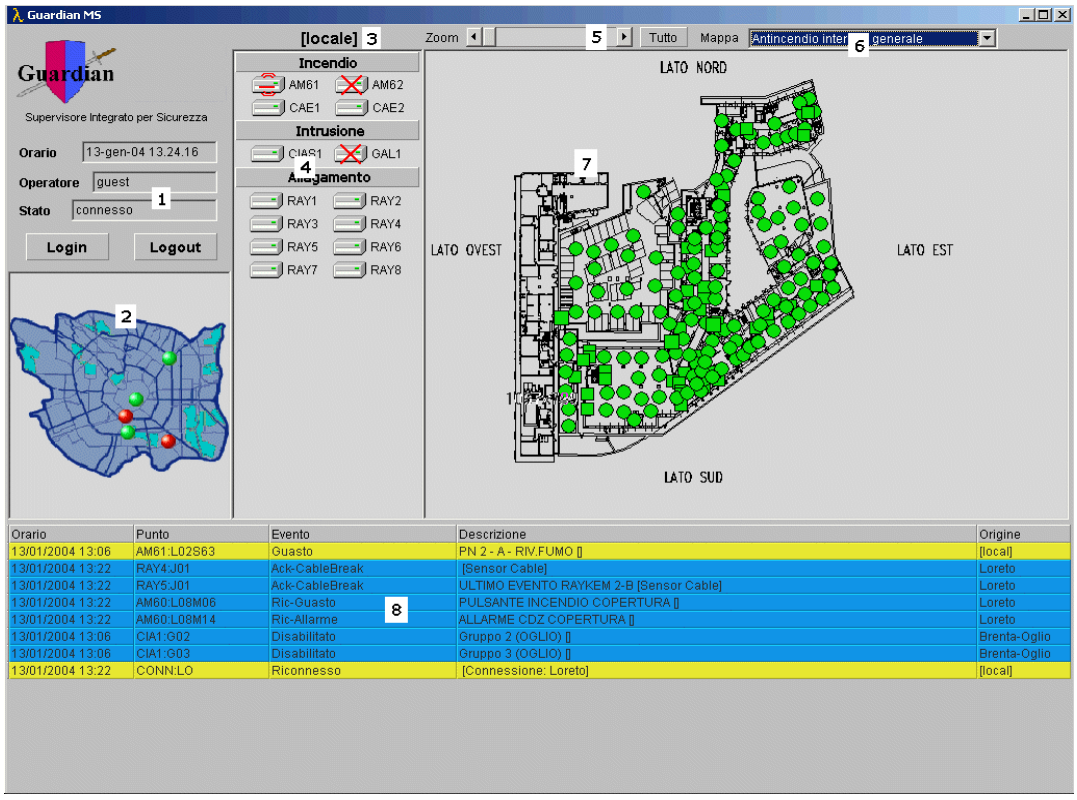
***Δ-VID***

Driver di ingresso video. Può acquisire direttamente telecamere BNC, USB oppure interfacciare video server o altri moduli Δ-VID. Il video viene riportato all'interno di Guardian permettendo una gestione combinata video-allarmi.

## Lambda: il client

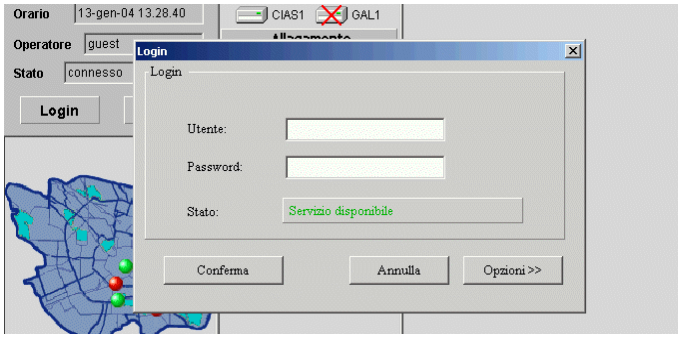
Il client di Guardian viene distribuito senza licenze e per una installazione di Guardian può essere installato un numero di client senza limitazioni.
Si tratta di client non specializzati, che recuperano cioè dal server tutta l'interfaccia operativa e tutte le funzionalità che forniscono.
Una installazione di Guardian viene quindi configurata e programmata interamente sul server.
I client possono essere ospitati su pagine HTML oppure funzionare come applicazioni autonome. Nel primo caso si dovrà avere l'accortezza anche di preparare alcune pagine web ospitanti, possibilmente integrate in un contesto più ampio.
Il disegno della interfaccia si basa su speciali files, gli LNI, che rendono particolarmente rapido l'adattamento del client alle esigenze del cliente. Per questo motivo ogni installazione di Guardian ha un layout differente con funzionalità differenti.
Nella descrizione seguente verranno utilizzate immagini relative ad una installazione MS di Guardian, cioè della versione MultiSite in cui vari server autonomi e limitati a specifici siti (edifici) vengono raccolti da un server centrale con layout differente, in una architettura NON simmetrica.

### Le parti del layout




### 1 – Informativa sulla connessione

Una speciale area sempre presente, anche se con diversa disposizione e grafica delle parti, visualizza poche fondamentali informazioni sulla connessione attuale. Viene indicato l'orario del server (a cui tutti gli orari degli eventi fanno riferimento), l'utente attualmente connesso e lo stato di connessione.
Fanno parte dell'area anche due pulsanti.
Il pulsante di **Login**, apre una finestra di dialogo in cui l'operatore inserisce il proprio nome utente e password e tramite la quale può opzionalmente cambiare la propria password.
Il pulsante di **Logout**, tramite il quale un operatore può terminare la propria sessione.




Nello schema di autenticazione di Guardian esiste sempre un utente connesso, che di solito è l'utente anonimo 'guest'. Tale utente può vedere gli allarmi e le mappe, ma non può riconoscerli o effettuare alcun'altra operazione. Quando un utente si collega, in base alla sua identificazione gli vengono forniti dei permessi (sule operazioni o sulla visibilità) e tutte le operazioni effettuate vengono registrate a suo nome. Per questo motivo se un operatore lascia la propria postazione, prima che un altro operatore effettui il login, è bene che effettui il logout, per tornare all'operatore di base 'guest' ed evitare di prendere la responsabilità di operazioni effettuate in sua assenza.

### 2 – Mappa comulativa

Una mappa raramente presente nelle versioni non MS, che può visualizzare una città, regione o l'intero stato, fornisce informazioni cumulative di allarme in sistemi distribuiti su ampia area geografica.
Ha il solo scopo di fornire un colpo d'occhio sulla situazione globale indicando quali siti hanno bisogno di attenzione.
Nella versione MS di Guardian questa mappa è sempre presente e permette il passaggio contestuale fra i server controllati.

### 3 – Indicatore di contesto (solo MS)

Indicatore del sito correntemente visualizzato. Ha senso solo in una architettura asimmetrica come MS.

### 4 – Lista centrali

Questo elemento è fondamentale e sempre presente.
Raccoglie tutti i dispositivi di campo controllati (centrali), in una struttura gerarchica organizzandoli per gruppo. Nella configurazione di Guardian sono associabili vari livelli di gruppo ad una centrale che a seconda delle preferenze possono riguardare la tipologia della centrale, la zona nella quale si trovano o una organizzazione interna del cliente.
Tramite questa lista possono essere gestite le centrali: cliccando su una di esse si aprirà infatti il menu di centrale


Questo menu varia fortemente da centrale a centrale e di solito contiene alcuni comandi globali, quelli da effettuare direttamente sulla centrale, più l'apertura di alcune tabelle di gestione che di solito riguardano la lista delle aree, la lista dei codici e la lista delle zone. In alcuni casi, per le centrali che lo consentono, sono presenti anche i pannelli di gestione delle uscite.
Come per tutte le altre operazioni in Guardian, anche per queste voci sono associabili dei livelli di accessibilità che inibiscono le funzionalità per operatori non autorizzati a quella specifica operazione.
La lista centrali inoltre riporta alcune fondamentali informazioni sullo stato della centrale, tramite semplici icone

- Centrale disconnessa (o non raggiungibile)
- Centrale in allarme
- Centrale in verifica (allineamento)
- Centrale in manutenzione
- Centrale occupata ((BUSY)
- Sospensione manuale della connessione

### 5 – Controllo ZOOM della mappa

Questo elemento è presente solo se si utilizzano mappe vettoriali, cioè convertite direttamente da disegni CAD, per le quali è sensato ingrandire o rimpicciolire l'immagine per identificare i dettagli con maggiore precisione.
Nel caso di mappe bitmap, l'utilizzo di questo controllo è sconsigliato per l'antiestetica perdità di definizione che ne consegue.

### 6 – Selettore di mappa

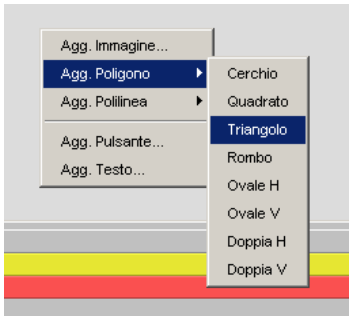
Indica sempre la mappa attualmente visualizzata e permette di passare ad altre mappe.
E' uno strumento non indispensabile, poiché nelle mappe possono essere inseriti elementi grafici di navigazione.

### 7 – Mappe

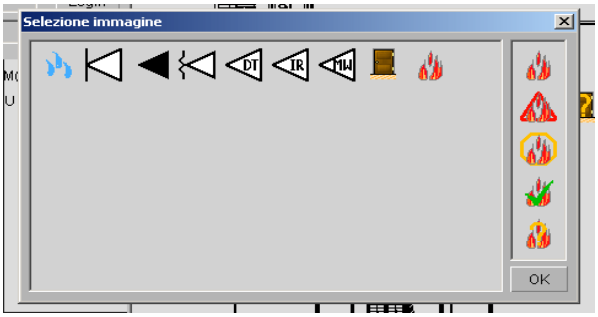
Le mappe di Guardian possono essere di due tipi, vettoriali o bitmap, utilizzabili anche in modo eterogeneo.
All'interno delle mappe vengono configurati alcuni elementi grafici di seguito elencati

- Marcatori poligonali: rappresentano un sensore tramite un poligono, cerchio o altro il quale assumerà un colore specifico a seconda dello stato del sensore.
- Marcatori bitmap: rappresentano un sensore tramite un insieme di immagini, ciascuna delle quali associata ad uno stato del sensore
- Marcatori polilineari: simili ai marcatori poligonali, ma definibili come una linea spezzata, per rappresentare elementi come cavi o barriere
- Elementi di testo: semplici elementi didascalici
- Pulsanti di navigazione: permettono la navigazione fra le mappe

Per inserire o configurare uno di questi elementi nelle mappe è necessario autenticarsi su un qualunque client con un utente particolare che abilita le funzioni di editing di mappa. Nel caso Guardian sia installato con la 'configurazione protetta', questo non sarà sufficiente e sarà necessario anche inserire nel computer sul quale gira il client anche una chiavetta HW specifica.
Quando un utente ha i diritti di modifica delle mappe, oltre alle ususali operazioni sulle mappe, ottiene anche l'accesso al menu di modifica e di aggiunta elementi

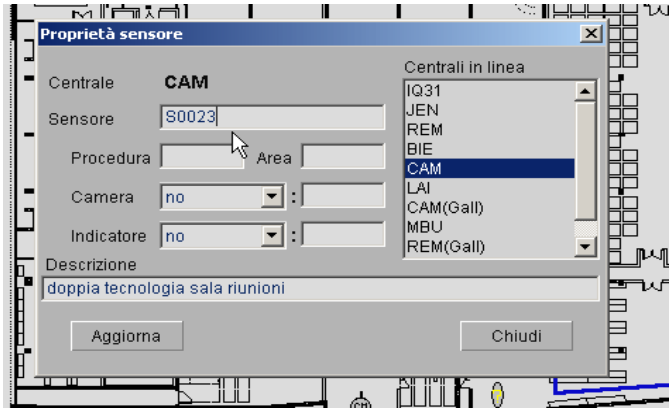



Il menu di aggiunta permette di aggiungere uno degli elementi precedentemente elencati. Nel caso di un marcatore bitmap (Aggiunta Immagine) si aprirà un ulteriore pannello che permette di scegliere le icone sensore da un set predefinito

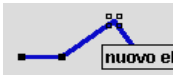



Tutti gli elementi aggiunti possono essere posizionati con precisione sulle mappe tramite semplice trascinamento (ovviamente solo per utente abilitato all'editing di mappe).
Il menu di modifica invece permette di cancellare o modificare le proprietà di un oggetto già inserito nelle mappe.

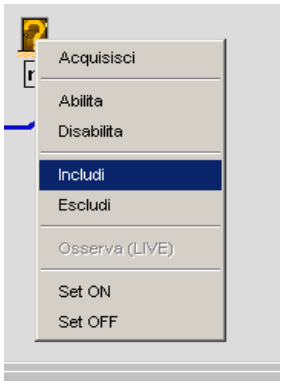

In particolare la voce proprietà apre il pannello di configurazione di un marcatore e ne permette l'associazione al sensore.




In questo pannello si seleziona una delle centrali configurate ed l'identificativo del sensore.
Altri dati sono la descrizione del sensore e la sua eventuale associazione ad una telecamera.
E' inoltre possibile associare una procedura ad un sensore. In questo caso su evento la procedura sarà richiamabile su richiesta dell'operatore.
Per i marcatori polilineari, oltre lo spostamento semplice sulla mappa è anche possibile il trascinamento dei singoli vertici, in modo da poter disporre la polilinea sulla mappa nel percorso più consono (molto comodo per i cavi antiallagamento e le barriere ad infrarossi)



Al di la delle operazioni di editing dei marcatori, sulle mappe possono essere effettuate una serie di operazioni da parte degli operatori autorizzati, come l'inclusione/esclusione dei sensori, l'inibizione/abilitazione degli stessi, il settaggio della uscita (se il marcatore viene associato ad una uscita) e il richiamo del video LIVE.




Per questa particolare funzionalità si possono predisporre anche marcatori con l'immagine di una telecamera, non associati ad alcun sensore, ma associati solo ad una telecamera nel caso l'unica funzione che si vuole ottenere sia appunto la richiesta del video LIVE.

## 8 – Sinottico eventi

La lista eventi contiene tutti gli eventi attivi (allarmi, guasti o segnalazioni) raccolti da Guardian.
Guardian prevede per ogni evento una delle 2 possibili transizioni
1 – **Evento** (ACK) **Evento riconosciuto** (END) **Riposo**

2 – **Evento** (END) **Evento terminato** (ACK) **Riposo**
In entrambe le sequenze l'iter dell'evento è determinato da 2 condizioni, l'ACK cioè l'azione di riconoscimento e presa in carico da parte dell'operatore, e dall' END, cioè la condizione, proveniente dal campo, di evento non più sussistente. Senza una delle due condizioni l'evento non può sparire dalla lista degli eventi (riposo).
I colori nella lista degli eventi aiutano a seguire l'evento nella sua evoluzione

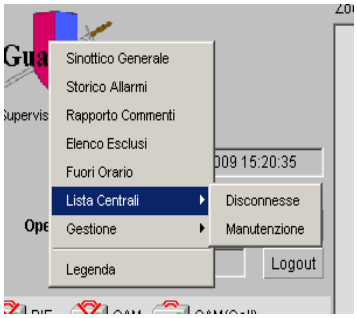
- Rosso – è un evento o un evento terminato (ma non riconosciuto) di tipo allarme.
- Giallo – è un evento o un evento terminato (ma non riconosciuto) di tipo guasto.
- Azzurro – è un evento riconosciuto di tipo allarme o guasto.

I colori dei sensori sulle mappe corrispondono a quelli degli eventi relativi nella tabella eventi, con la sola eccezione che nelle mappe i sensori non riconosciuti in allarme o in guasto possono essere resi lampeggianti per una più facile localizzazione.


Oltre al comando di riconoscimento, l'altro fondamentale comando sulla lista eventi è la localizzazione.
Tramite questo comando Guardian cambierà la mappa posizionandosi su quella che contiene il sensore relativo all'evento selezionato.
In Guardian non sono previsti cambi di mappa automatici su arrivo di eventi: questa è una precisa scelta in quanto all'arrivo di un evento l'operatore potrebbe essere impegnato su una mappa e sarebbe poco funzionale toglierla dalla sua vista in modo arbitrario.
Se il sensore relativo all'evento è associato ad una telecamera, da questo menu possono essere richiamati anche i comandi Osserva LIVE e Osserva EVENTO. Il primo aprirà un popup del LIVE della telecamera associata, mentre il secondo un popup visualizzante la registrazione di quella telecamera a partire da 10 secondi (tempo configurabile, ma globalmente, non per singolo sensore) prima dell'orario dell'evento, in modo da avere una panoramica temporale attorno all'evento.
Guardian si occupa di effettuare tutte le conversioni di orario necessarie, rendendo superfluo l'allineamento dell'orario di tutti i videoserver.

## Il menu generale

Selezionando lo scudo di Guardian, in alto a sinistra del client, si apre il menu generale che permette l'accesso a delle funzioni avanzate di configurazione, reportistica e altro.



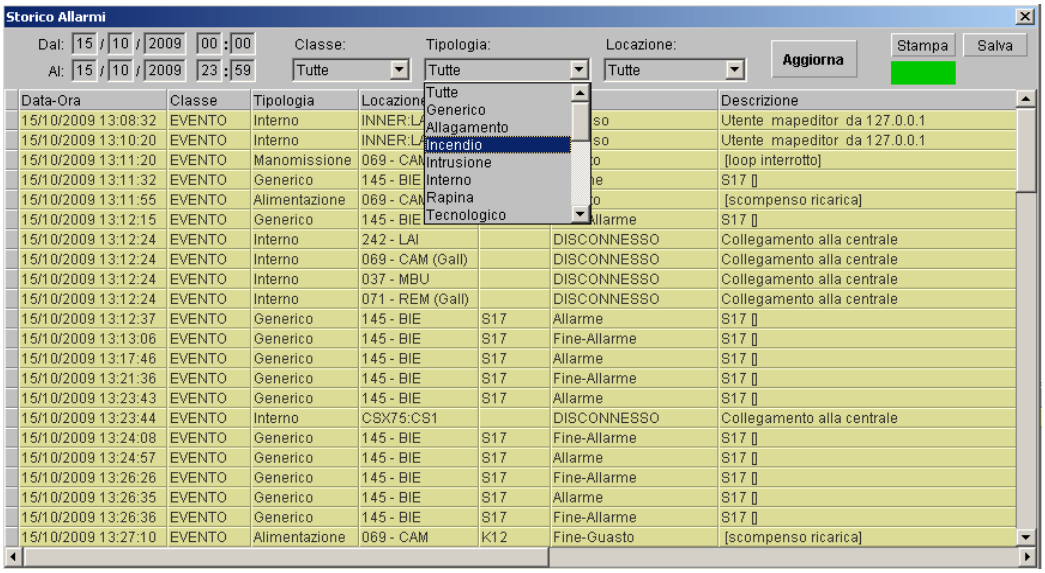

**Sinottico generale**

E' un pannello che riassume, in formato grafico, tutte le aree presenti nel sistema, e mostra se sono inserite (rosse) o disinserite (verdi).
Questo pannello viene spesso inserito come una sezione del layout di Lambda, direttamente senza doverlo aprire, nel qual caso questa opzione di menu diviene superflua e viene tolta.

**Storico allarmi**

E' una pannello che permette il recupero degli eventi storicizzati tramite 4 filtri associati:

- selezione temporale
- classe evento (evento, azione operatore o stato)
- tipologia dell'evento (le tipologie sono associabili liberamente ai sensori)
- locazione (sfrutta i gruppi associati alla centrale a cui l'evento fa capo, ad esempio la sede)




La funzione di stampa, ampiamente configurabile, permette la generazione di report graficamente gradevoli su carta.
Tramite questo strumento è possibile tracciare tutta la 'vita' di un evento, valutare se è stato preso in carico dall'operatore in tempi accettabili, ecc.

**Rapporto commenti**

Ogni operatore può associare un commento ad un evento quando lo prende in carico.
Il pannello di rapporto commenti permette di visualizzare i commenti immessi dagli operatori, sostituendo brevi note cartacee (post-it)

**Elenco esclusi**

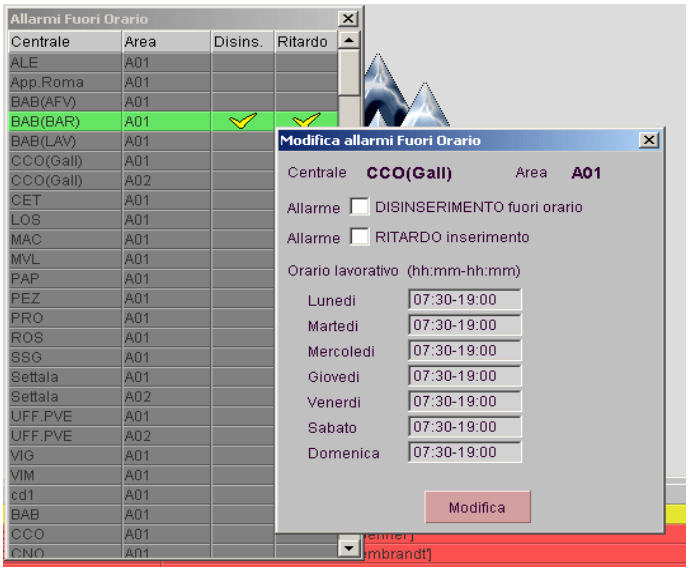
Un pannello sinottico che riassume tutti i sensori attualmente esclusi o inibiti nel sistema, fornendo informazioni sull'orario di esclusione.
Il pannello, che fornisce filtri simili allo storico eventi (tranne quello temporale), è uno strumento utile per non dimenticare sensori esclusi.

**Fuori orario**

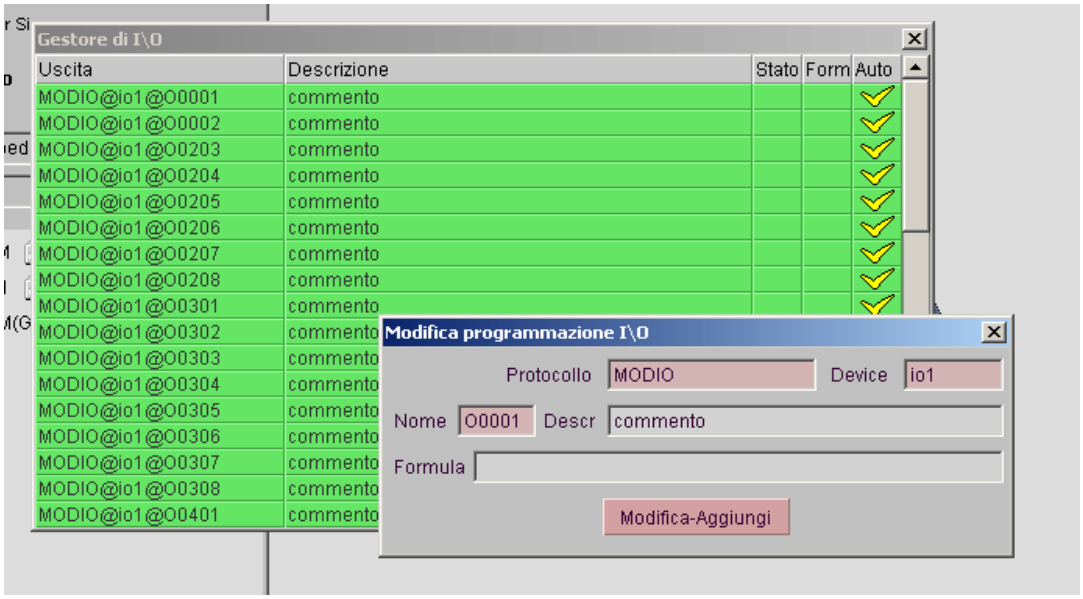
Per ogni singola area, Guardian permette di impostare delle segnalazioni di anomalie rispetto ad una schedulazione settimanale.
Tramite il pannello dei fuori orario si può configurare per ogni giorno della settimana un intervallo orario nel quale ci si aspetta che l'area sia inserita o disinserita e attivare gli allarmi di

- DISINSERIMENTO fuori orario – Se l'area viene disinserita da tastiera (non da Guardian) in un orario in cui ci si aspettava che l'area fosse inserita.
- RITARDO inserimento – Se arriva il momento di inserire le aree, ma l'operatore se ne dimentica, riceverà degli allarmi di ritardo inserimento, a promemoria.

Le fascie orarie all'interno di una giornata possono comprendere più intervalli.
Tutte le aree presenti nel sistema (appartenenti a tutte le centrali antintrusione), rientrano nella lista presentata dal pannello, anche se inizialmente per tutte i due tipi di allarme saranno disattivati, fino a differente programmazione esplicita.




**Lista centrali – Disconnesse**
Un semplice sinottico che riassume tutte le centrali attualmente non raggiungibili. Utile solo nel caso di sistemi enormi.
**Lista centrali – Manutenzione**
Tramite apposito comando sulla centrale, una centrale può essere posta in stato di manutenzione, per cui Guardian ignorerà tutti gli eventi provenienti da essa, finché non verrà di nuovo rimessa in condizione normale.
Questo sinottico riassume le centrali attualmente poste in tale stato.
**Gestione – Input/Output**
Il gestore di IO di Guardian permette di configurare formule booleane complesse per pilotare qualunque uscita collegata al sistema.




Le uscite pilotabili non sono solo quelle dei device di IO, ma anche quelle delle centrali di sicurezza collegate.
Possono essere inseriti nelle formule booleane i seguenti elementi

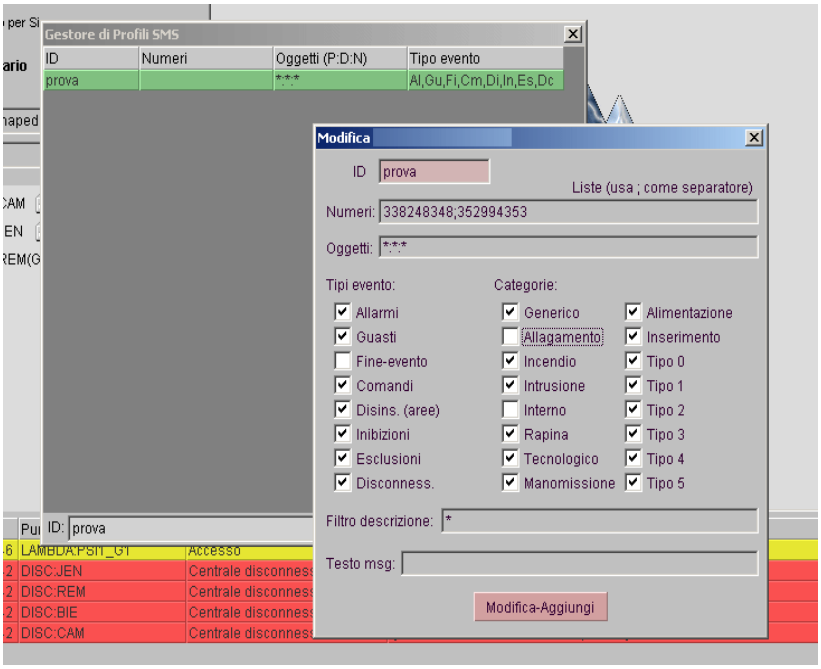
- allarmi di sensori
- guasti di sensori o di sistema
- stato inserimento aree
- allarme di area

Il formalismo di tali formule esula dallo scopo del presente documento.
**Gestione – Sorgenti video**

E' una sorta di anagrafica dei videoserver di cui Guardian necessita per poi permettere l'associazione delle telecamere ai sensori.
In questa anagrafica vengono inserite anche le credenziali per l'accesso ai videoserver, e le tipologie di popup da utilizzare.

**Gestione – Profili SMS**

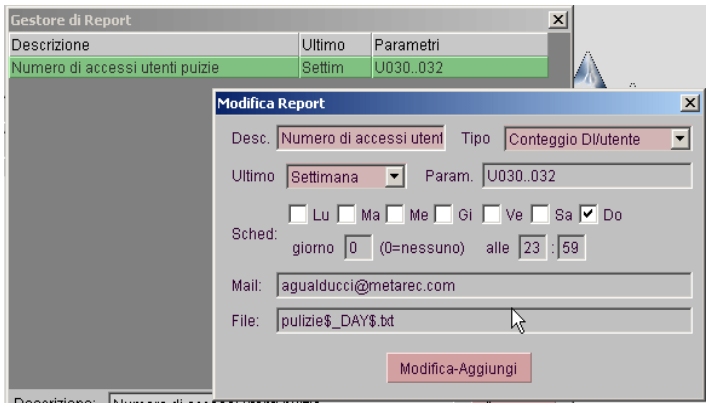
Permette di creare una lista di profili di notifica e per ciascun profilo configurare un insieme di numeri telefonici di destinazione, una serie di tipologie di allarmi da notificare e un formato di messaggio da inoltrare.




La notifica SMS per funzionare deve far capo ad un dispositivo MC55 (modem GSM) che deve essere collegato e raggiungibile dal server di Guardian.

**Gestione – Report**

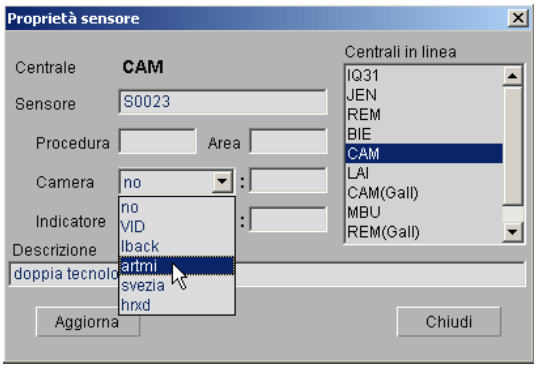
I Report Parametrici di Guardian possono essere definiti per produrre un'ampia gamma di reportistica attivabile con schedulatore (giornalieri, settimanali, mensili, annuali) o manualmente.




Questi report attingono alla stessa base dati dello storico allarmi, ma la sintesi di cui sono capaci permette di estrarre informazione comoda e con un valore aggiunto.
Possono essere salvati su file, inviati per posta o entrambi.

## Integrazione eventi-video: i popup

Guardian permette una efficace e duttile integrazione fra eventi di sicurezza (intrusione, incendio, allagamento) o tecnologici (stati e soglie) e le sorgenti video configurate.
Questo viene realizzato tramite dei popup, cioè finestre in primo piano che visualizzano sempre un solo flusso video e non hanno interazione con l'utente se non lo spostamento e la chiusura del popup stesso.
Qualunque marcatore grafico nelle mappe può essere associato ad una sorgente video, ma una sola, tramite il pannello di proprietà del marcatore.




Al marcatore si associa la sorgente video (videoserver) ed i parametri, come nella immagine sopra.
I parametri sono della forma: N$Pxxx$Pxxx$Pxxx...
dove N è il numero di telecamera, relativamente al server video indicato, mentre i parametri sono tutti opzionali e sono divisi in 2 categorie:

**Parametri interni**

Sono parametri utilizzati dal client Lambda e quindi validi per tutti i popup.

**$A** – Attiva AUTOPOP, cioè all'arrivo di un allarme sul sensore associato al marcatore, viene aperto il popup video. A seconda della installazione, il popup può partire in LIVE (video corrente), oppure può partire in GET (video registrato) a partire da un certo numero di secondi precedenti l'allarme stesso.

**$AL** – Variante di $A da utilizzare nel caso di popup che partono in GET. Permette di stabilire, per un certo sensore-marcatore, che su allarme il suo video deve partire in LIVE, invece che in GET.

**$AH** – Variante di $A per far partire il popup video nascosto (non visibile). Un popup ovviamente non visualizza alcun video, ma può impostare dei preset sulla telecamera.

**$LB*source*** – Imposta Live BackDrop su quel marcatore. Ogni volta che questo popup viene fatto partire in LIVE, viene utilizzata la sorgente video (videoserver) indicato da source, al posto della sorgente video impostata nel pannello di preferenze del marcatore. Questo permette, nel caso di videoserver con telecamere IP, di bypassare il videoserver stesso per ottenere il flusso video in LIVE, accedendo direttamente alla telecamera.

**Parametri esterni**

Sono parametri interpretati dal popup specifico e possono essere o non essere presenti, a seconda delle funzionalità implementate. Di norma questi parametri definiscono la qualità del video e funzioni PTZ.
Descritti di seguito solo alcuni disponibili in molti casi: per gli altri fare riferimento alla documentazione specifica dei popup.

**$P*nn*** – Imposta sulla telecamera il preset numero nn (due cifre, ad esempio '$P02' per il preset numero 2).

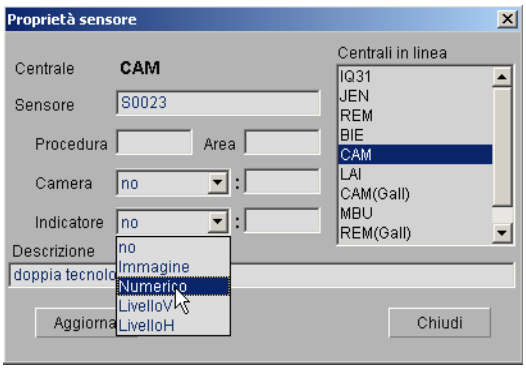
**$B** – Attiva la gestione del brandeggio sul popup.

Per quanto riguarda l'autopop (attivazione automatica dei popup su allarme), ciascuna installazione di Guardian può definire una particolare politica di allocazione dei popup sullo schermo, ad esempio una allocazione a cascata, oppure tutti su un lato, su una riga, ecc. Sta comunque all'utente chiudere i popup quando non servono più.
Le altre modalità di accesso ai popup sono:

- Richiesta LIVE da mappa
- Richiesta LIVE dalla lista degli eventi correnti
- Richiesta GET (n secondi prima) dalla lista degli eventi correnti
- Richiesta GET (n secondi prima) dallo storico allarmi

## Integrazione regolazione meccanica: i gauge

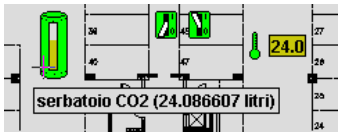
Guardian integra le segnalazioni di allarme con la gestione dei valori di campo tipica della automazione industriale e della regolazione meccanica. Per fare questo si utilizzano degli elementi grafici in sovrapposizione ai classici marcatori chiamati 'gauge' (indicatori).




Ciascun elemento di campo può avere, in aggiunta al suo stato di allarme, anche i seguenti 3 elementi:
- valore di campo
- valori di range (min,max)

Esistono 4 tipi di gauge:

- **Immagine** – ottenuto sovrapponendo delle icone descrittive ai marcatori di mappa. Tale indicatore è adatto a valori 0,1 (digitali) o comunque a valori finiti e limitati – NON necessita del range
- **Numerico** – sovrappone al marcatore un riquadro che visualizza il valore di campo – NON necessita del range
- **LivelloV** e **LivelloH** – sovrappone al marcatore una barra di riempimento orizzontale o verticale ed è adatto a visualizzare un riempimento di un serbatoio, la carica di una batteria, o la percentuale di completamento di un processo. NECESSITA del range.



-esempi di gauge-

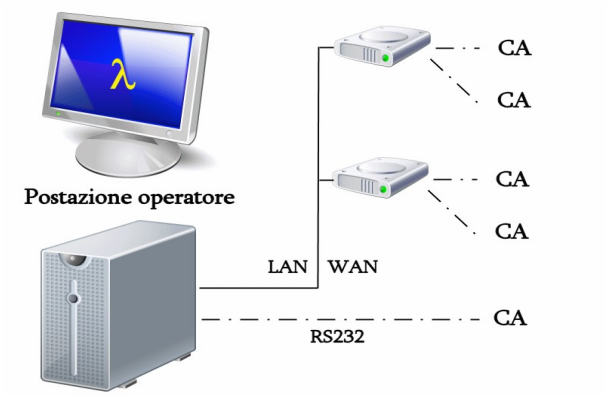
Tutti i driver di Guardian sono in grado di produrre i valori di campo per gli elementi che controllano: i gauge immagine possono essere quindi utilizzati in tutti i casi, anche solo per segnalare i sensori attivi sulla mappa o per evidenziare accessi (barriere, varchi) aperti, sia sui sensori, sia sugli I/O di centrale.
I gauge numerici e di livello sono invece utili solamente per i dispositivi specifici per l'automazione.
Ciascun gauge permette inoltre di alterare i valori di campo con delle scritture. A seconda delle situazioni questo può essere fatto anche con pulsanti specifici preimpostati, oppure con i comandi da menu.

## Topologia

L'architettura del server di Guardian sfrutta una modularità che gli permette di essere installato in varie configurazioni che seguono al meglio le esigenze del cliente.
Il server infatti può risiedere interamente su una singola macchina fisica (PC server) oppure essere distribuito su diverse macchine.
Di seguito vengono illustrate alcune applicazioni tipiche, attive presso clienti.

### Installazione a macchina singola

Si tratta della installazione più naturale ed immediata in cui un unico server fisico gestisce in modo autonomo tutte le componenti del supervisore.



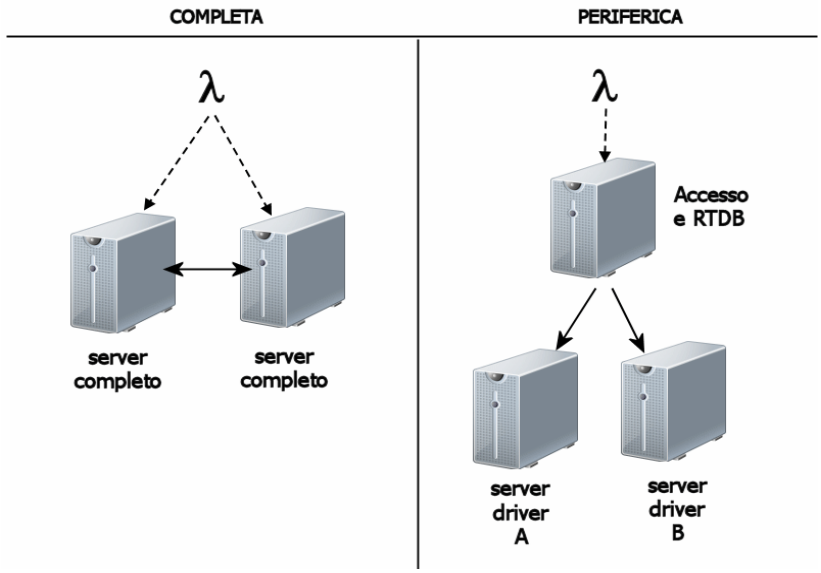

Il server di Guardian, compresi i driver per i vari protocolli, possono essere ospitati direttamente sulla postazione dell'operatore, su cui funziona anche il client Lambda. Le centrali d'allarme (CA) vengono raggiunte direttamente tramite seriale a bordo macchina, oppure tramite serial-hub dislocato in remoto e quindi via LAN o WAN (cioè con o senza routing).
Anche in questo caso non ci sono limitazioni al numero di client collegabili.
Le installazioni di questo tipo risultano poco costose, efficaci, facili da manutenere, facilmente estendibili.
Si consiglia comunque, per motivi di sicurezza e di stabilità, di non utilizzare il client sulla stessa macchina del server che in generale dovrebbe essere una macchina cieca (priva di monitor e tastiera) e di utilizzare come client sempre dei computer collegati in rete, di classe desktop.

### Installazione in backup

Per assicurare una maggiore continuità di servizio nel caso di cedimenti hardware è possibile configurare due server fisici di Guardian con gli stessi moduli e la stessa configurazione per funzionare in backup caldo fra di loro.
In questo caso ci sono diverse soluzioni di ridondanza possibili:

- ridondanza completa – l'intero server (driver compresi) è ridondato
- ridondanza periferica – solo le macchina adibite ai driver vengono ridondate
- ridondanza mista – comprende tutti gli schemi intermedi
-

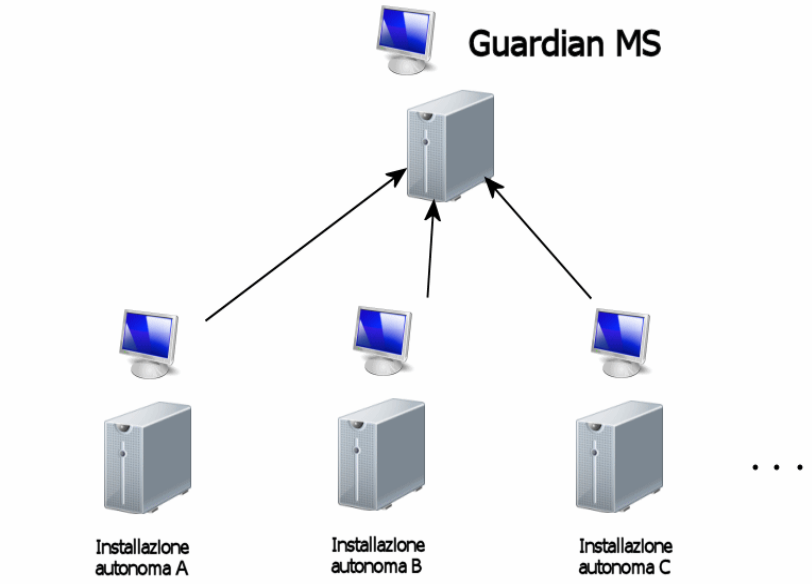
Nella **ridondanza completa** l'accesso al campo (centrali d'allarme) è di tipo concorrente e i due server competono per l'acquisizione delle risorse non condivisibili, tra cui tipicamente l'accesso alle seriali a bordo macchina ed alle porte server dei serial-hub. L'allineamento fra i due RTDB (parti centrali dei server di Guardian) viene assicurato dal 'double-feed' dei driver, cioè tutti i driver mandano gli stati del campo ad entrambe i server, e i comandi verso il campo parimente vengono spediti a tutti i driver. Si tratta quindi di un allineamento passivo.
I client installati in rete devono essere configurati per accedere ad entrambi i server: utilizzeranno poi uno o l'altro server a seconda di quale è attivo.




Nella **ridondanza periferica** invece, utilizzata anche per estendere la capacità di supervisione di Guardian, il server centrale (RTDB) utilizza i driver situati su server periferici in modo selettivo e quindi l'allineamento è non necessario. Non c'è accesso concorrente al campo.
Con queste due architetture di backup sono ad oggi gestiti diversi impianti su scala nazionale con oltre 200 centrali ciascuno e diverse peculiarità operative.

### Installazione asimmetrica (Guardian MS)

La sigla MS della versione di Guardian significa multi-site.
Ma questo non ha nulla a che fare col numero di computer su cui Guardian è installato, né con l'ampiezza dell'installazione.
In generale Guardian MS può essere visto come un ulteriore strato di supervisione posto sopra un certo numero di installazioni di Guardian **autonome e indipendenti**.




Le installazioni autonome possono mostrare fra di loro differenze notevoli, nel tipo e numero di mappe, nelle centrali configurate, negli operatori configurati, ecc., ma soprattutto nel layout del client, cioè nella disposizione degli oggetti di gestione dell'impianto.

Guardian MS si collega a tutti i siti periferici e ne ingloba la configurazione, predisponendo un sovra-strato di supervisione basato sul cambio di contesto, da cui il nome MultiSite.
Tutti i controlli tipici di Guardian, in Guardian MS tengono conto dei contesti: ad esempio nello Storico Allarmi la usuale griglia contiene una colonna in più (e quindi anche un filtro in più) che specifica il contesto dell'evento.
Nota: un server di tipo MS ingloba SEMPRE anche un server normale, che supervisiona le centrali locali e costituisce in Guardian MS un contesto sempre presente, chiamato appunto 'locale'.

## Allegati

**Protocollo REX**: descrizione del protocollo interno fra i moduli del server
**Protocollo InterCOM**: descrizione del protocollo di distribuzione verso Lambda

## Protocollo REX

Il sistema di comunicazione REX è stato ideato per sostituire il bus informativo RMI, mantenendone molte caratteristiche, come il fatto di essere un RPC e di basarsi su un Naming Service (RA), ma con caratteristiche che lo differenziano.

- I metodi inclusi nell'interfaccia non sono definibili, ma sono calibrati sulle necessità di comunicazione dei moduli di JaMIX (in particolare col server **Ψ)**
- Le performance sono nettamente migliori di RMI (oltre 3 volte più veloce)
- E' una interfaccia BIDIREZIONALE e simmetrica (il client espone sempre gli stessi metodi del server). Per ottenere la stessa cosa con RMI era necessario che entrambi i moduli fossero sia client che server.

### Il package cost.util.rex

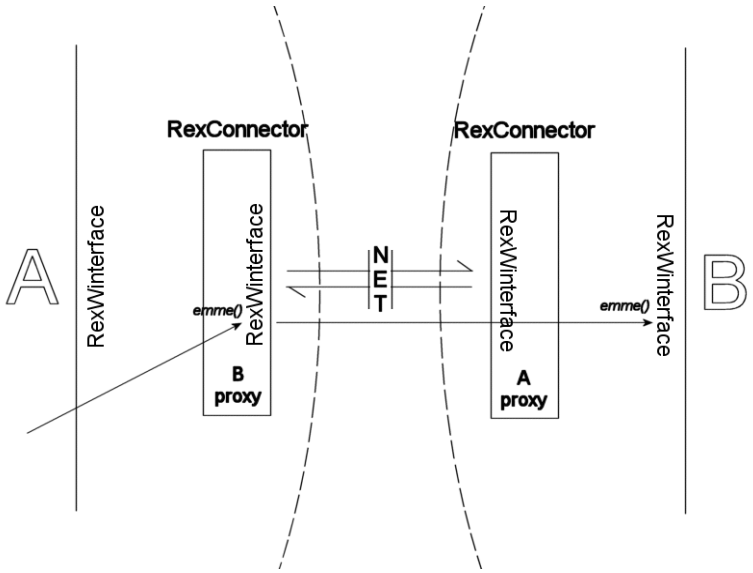
REX è completamente contenuto nel package **cost.util.rex**
Tale package contiene solo 4 classi

- RexWinterface
- RexConnector
- RexServer
- RexServerData

Le prime due classi, *RexWinterface* e *RexConnector*, servono a qualunque modulo voglia comunicare via REX, sia esso client o server. Le altre due classi servono solo ai moduli che agiscono da server per REX (in JaMIX solo il modulo **Ψ** è server, tutti gli altri si connettono ad esso).

### Schema di utilizzo

Ogni modulo che voglia utilizzare REX per comunicare con il modulo **Ψ** di JaMIX deve fare 2 cose
**Implementare** l'interfaccia *RexWinterface*
**Istanziare** un *RexConnector*




L'istanza di *RexConnector* implementa a sua volta l'interfaccia *RexWinterface* e, opportunamente configurata, funziona da proxy per la controparte.
Nello schema in figura sono presenti due moduli, A e B i quali potrebbero essere verosimilmente: A un modulo del cluster che voglia connettersi (client) a **Ψ**, e B il modulo **Ψ** stesso.
Se A vuole richiamare un metodo *emme()* sul modulo B, non dovrà fare altro che richiamarlo sulla **interfaccia del connettore** allocato localmente che agisce da proxy per B. Il connettore invierà la chiamata in modo trasparente **alla interfaccia interna** del modulo B. Tale meccanismo in figura è illustrato dalle frecce. Il valore di ritorno del metodo, eseguito su B, segue esattamente la stessa strada a ritroso.
Se viceversa il modulo B vuole richiamare un metodo su A, dovrà effettuare una chiamata sulla interfaccia del suo connettore locale che agisce da proxy per A e il connettore provvederà a richiamare il metodo sulla interfaccia interna di A: lo schema quindi è completamente simmetrico ed entrambi i moduli possono richiamare metodi uno sull'altro.

### RexWinterface

Descriviamo ora i metodi dell'interfaccia

public boolean **RW_Init**(String wname, RexConnector rconn, Object[] args);
public boolean **RW_Finish**();
public boolean **CreateTAG**(String tagName, Template tout, String classname, byte[] codice, int clear, long area, String args);
public boolean **SetTAGSource**(String tagName, String tagFrom);
public Template **InputTAG**(String tagName);
public boolean **OutputTAG**(String tagName);
public String[] **PutRecord**(Record[] reca);
public int[] **GetStatus**();

I metodi RW_Init() e RW_Finish() non sono propriamente metodi remotizzati, ma sono utilizzati per la gestione dei connettori e delle classi di servizio.
Questi due metodi sono quindi importanti per l'inizializzazione della connessione REX e per la sua chiusura.
Vedremo nel prossimo paragrafo come un client può inizializzare una connessione verso **Ψ**.
In generale una classe client che voglia implementare l'interfaccia *RexWinterface* potrà lasciare vuoti questi metodi.

```
public boolean RW_Init(String wname, RexConnector rconn, Object[] args)
{
	return true;		// Fittizio
}
public boolean RW_Finish()
{
	return true;		// Fittizio
}
```

Il metodo **CreateTAG()** permette di definire una TAG all'interno di **Ψ** e tale metodo non verrà **mai** richiamato sul client. **Ψ** è infatti l'unico modulo che può ospitare TAG. Ci sono varie modalità di creazione delle TAG in funzione dei parametri passati.
Se il parametro *classname* è una stringa vuota verrà allocata una TAGbare con struttura definita dal parametro *template*; se *classname* inizia con * verrà allocata una TAG di sistema il cui identificativo segue l'asterisco; altrimenti verrà utilizzato il parametro *codice* per definire una TAG e *classname* sarà il nome di classe associato.
I moduli che necessitano di inserire dati in **Ψ** solitamente utilizzano il primo sistema, mentre il **feeder** (cost.util.PSIfeed) utilizza alternativamente gli altri due.
Il metodo **SetTAGSource()** permette di collegare il flusso fra 2 TAG. Chiaramente può essere richiamato solo sul server **Ψ**.
Il metodo **InputTAG()** permette ad un modulo client di sottoscrivere sul server **Ψ** il flusso di una TAG, il cui nome è specificato in argomento.
Il metodo **OutputTAG()** permette ad un modulo client di dichiarare sul server **Ψ** una TAG di destinazione, il cui nome è specificato in argomento.
Il metodo **PutRecord()** permette il trasferimento dei record da server a client e viceversa. Nessun modulo può dare per scontato di non ricevere chiamate su questo metodo, anche se non sottoscrive nulla. Tuttavia può decidere di ignorare i record ricevuti, ovviamente.
Nella implementazione di PutRecord() si consiglia di inviare, laddove è possibile, i record in array di dimensione maggiore di 1 per aumentare il throughput di REX. Questa tuttavia è solo una indicazione generica, infatti le chiamate REX non sono molto costose, ma potrebbe esserlo il frazionamento dei pacchetti TCP.
Il metodo ritorna una stringa per ogni record, recante, in sequenza, la risposta all'inoltro del record.
Il metodo **GetStatus**() serve per fini diagnostici, di controllo, ecc.
La sua definizione non è ancora pronta alla stesura di questo documento. Una risposta di rifiuto è comunque utilizzabile

```
public int[] GetStatus()
{
	return null;		// Fittizio
}
```

### RexConnector

E' la classe che implementa effettivamente tutto il lavoro di remotizzazione e si occupa del look up del server, della riconnessione, della sincronizzazione delle chiamate, ecc.

La cosa principale da sapere è come inizializzare il connettore.
Vediamo il seguente esempio, relativo ad un modulo client (rispetto al server **Ψ**)

```
RexConnector rexconn = new RexConnector(url, rwi, uninumber, name);
```

Descriviamo quindi il significato dei parametri utilizzati:
**url** è una string della forma "rex://host/modname". Un eventuale numero di porta dopo il nome dell'host verrà ignorato. *modname* rappresenta il nome con cui il server si è pubblicato (tipicamente 'PSI1'). Il protocollo può essere implicito, cioè "//host/modname" è altresì valido.
Il costruttore di RexConnector applica la risoluzione delle variabili distribuite di RA alla stringa url: quindi tipicamente il nome dell'host può essere rappresentato dal nome di una variabile distribuita, racchiuso da $…$
Esempio: `"rex://$HOST1$/PSI1"`
**rwi** è una istanza di una classe che implementi la *RexWinterface*. Su tale istanza verranno richiamati i metodi delle chiamate provenienti dalla controparte.
**uninumber** è il numero unico per il cluster. Se tale numero non corrisponde fra i moduli, la richiesta viene rifiutata e la comunicazione non avrà luogo.
**name** è la stringa con cui il modulo client vuole identificarsi. In alcuni casi può essere una sottoparte del modulo ad agire da client, quindi tale nome può essere diverso da quello del modulo.
Appena inizializzato il connettore è possibile utilizzare i metodi remoti. Tuttavia è necessario fare attenzione al fatto che il connettore lato client blocca i metodi finché non riesce ad eseguirli. Se una connessione risulta impossibile, il thread chiamante rimane appeso all'infinito: è bene quindi disaccoppiare esternamente la chiamata dei metodi dalle operazioni interne del modulo. Un esempio di disaccoppiamento si trova in **cost.tau.TAUPSILink**

### Ragent, il nameserver

Nella versione 33 di RA è stato inserito il comando NAMESERV per supportare REX. Questo significa che sarà necessario l'upgrade di RA a tale versione.
Tramite il sottocomando PUT un modulo server può pubblicare se stesso, ricevendo una porta server.
Tramite il sottocomando GET un modulo client può richiedere la porta server sulla quale si è attestato un modulo server e quindi raggiungerlo.
RA può limitare il numero delle porte server ad un certo range, permettendo di configurare il firewall lato server in modo da permettere l'utilizzo di REX, senza aprire tutte le porte.
Le connessioni di REX sono molto più statiche di quelle dell'RMI e non consumano le risorse di sistema in modo non prevedibile.
La connessione fra due moduli è sempre una sola socket, nella quale vengono multiplexate le chiamate asincrone.

## Protocollo InterCOM

La connessione diretta al **server periferico PHI** è da considerarsi una connessione *eccezionale* in quanto non segue lo schema di validazione utente di JaMIX.
La validazione viene effettuata localmente dal server periferico che si avvale anche di una white-list al fine di rendere la connessione sicura.
Tale accesso è riservato per sua natura a (pochi) utenti particolari che necessitino di rapidità nelle procedure di accesso.
La connessione potrà essere effettuata sia in modalità *ascii* che *binaria*, ma solo la prima sarà di fatti utilizzabile se il client non è sviluppato in Java.

Nel caso di una connessione *ascii* tutti i dati (tranne quelli dell'header) vengono trasferiti come *tipi base* la cui forma è la seguente: **Xaaaa\r\n** ove 'X' è un singolo carattere indicante il tipo semantico, 'aaaa' è il dato in formato stringa e '\r\n' sono i caratteri ascii 0x0D e 0x0A (CR+LF).
I caratteri indicanti il tipo semantico sono:

- **T** = string (es. "Ttesto inviato\r\n")
- **B** = boolean (es. "B0\r\n")
- **S** = short (16bit) (es. "S3865\r\n")
- **I** = int (32bit) (es. "I684932\r\n")
- **L** = long (64bit) (es. "L58938539845\r\n")
- **F** = float (32bit) (es. "F46.62\r\n")
- **D** = double (64bit) (es. "D349.039343\r\n")
- **Y** = byte (8bit) (es. "Y16\r\n")

Per accedere il client deve possedere le seguenti informazioni:

- Nome dell'host su cui si trova il server
- Porta TCP di ascolto del server
- Nome dell'utente (account): è una stringa alfanumerica non contenente spazi
- Chiave di accesso (password): è un numero negativo(solo cifre)

**Nota**: Le chiavi locali sono **sempre numeri negativi**

### Header

L'header utilizzato per la connessione simula l'header HTTP che costituisce il tipo di accesso primario al cluster di JaMIX.
L'header HTTP è composto, in generale, nel seguente modo:

**metodo /oggetto protocollo\r\n\r\n**

Nel caso della connessione a chiavi locali tale header viene utilizzato in modo improprio per fornire informazioni sull'utente e sullo scopo dell'apertura della connessione.
Tali informazioni tuttavia servono al server solo per richiamare la chiave locale associata a quell'utente, e non ancora a verificare l'identità dello stesso

### Creazione di una connessione

Per aprire una connessione dati utilizzare il metodo "CREATE" , inserire il nome dell'utente come oggetto e la chiave di accesso negativizzata come protocollo

**CREATE /myname -7854972\r\n\r\n**

### Recovery di una connessione precedente

Per ricoverare una connessione precedente utilizzare il metodo " RECOVERY" , inserire il nome del *gate* ottenuto dalla connessione precedente come oggetto e la chiave di accesso come protocollo

**RECOVERY /PHI01:G31 -7854972\r\n\r\n**

### Verifica utente

Da questo punto in poi tutti i dati inviati e ricevuti saranno codificati come *tipi base*.
La verifica utente viene fatta semplicemente inviando l'account e la chiave (che è un intero a 64bit)

**Tmyname\r\n**
**L-7854972\r\n**

### Specifiche di canale

E' una singola stringa che specifica il tipo di canale richiesto, la direzione dei dati e le caratteristiche operative
La string ha la forma "*Dn|Ts|Cx|bbbbb*" ove
'*D*' è la direzione dei dati ('I' = input, 'O' = output, 'B' = both)

‘*n*’ è il numero di canale (normalmente 0)
‘*T*’ è il tipo di canale (‘C’ = chameleon, ‘S’ = socket, ‘R’ = RMI, ecc.)
‘*s*’ sono opzionali parametri specifici dipendenti dal tipo di canale
‘*C*’ è la compressione richiesta (‘N’ = nessuna, ‘D’ = delta compression)
‘*x*’ sono opzionali parametri specifici dipendenti dal tipo di compressione
‘*bbbbb*’ è una stringa di zeri e uno che definiscono posizionalmente alcuni flag

- SYNCH: canale sincronizzato
- ALIVE: richiesto messaggio di alive sul canale
- CRYPTO: canale criptato
- RECOVERY: abilita la recovery sul canale
- RESEND: abilita il buffer di resend sul canale

Ad esempio

**TB0|Sa|D10|01011\r\n**

richiede il canale 0 bidirezionale (B0) di tipo Socket con dati in formato *ascii* (Sa) con compressione *delta* di profondità 10 per i dati applicativi (D10), non sincronizzato, con messaggi di alive, non criptato e ricoverabile (01011).
Questo esempio è anche la specifica consigliata per client non-Java.

Ricapitolando: fino a questo momento una connessione diretta può essere aperta inviando sulla socket il seguente buffer di esempio:

**CREATE /myname -7854972\r\n\r\n**
**Tmyname\r\n**
**L-7854972\r\n**
**TB0|Sa|D10|01011\r\n**

## Il record

Da qui in avanti la comunicazione prosegue con uno scambio di record tra il client ed il server.
Alcuni record saranno record applicativi, altri saranno record di servizio (META record).
Tutti però hanno la stessa struttura (è indicato fra parentesi il tipo base corrispondente):

**Flags** (**I**)
**TAGname** (**T**)
**TimeStamp** (**L**)
**SID** (**I**)
**SeqNo** (**I**)
**Size** (**I**)
**Field 1** (**T**)
**Field 2** (**T**)
……. …
**Field n** (**T**)

I **Flags** costituiscono una mappa di bit che fornisce caratteristiche del record, alcune interessanti per il protocollo, altre solamente applicative e non pertinenti con questo documento

| | |
|---|---|
| int FL_ALIVE = 1 | (applicativo) |
| int FL_SUBST = 2 | Il record ne sostituisce uno precedente con la stessa chiave |
| int FL_FIELD = 4 | (applicativo) |
| int FL_TRANS = 8 | (applicativo) |
| int FL_ADDED = 16 | Il record è in aggiunta (è il primo con la sua chiave) |
| int FL_REQAN = 32 | Il record richiede un record di risposta sul canale |
| int FL_DELTA = 64 | Il record è *delta*-compresso |

Il **TAGname** indica a quale TAG è associato il record. La TAG in JaMIX è un repository nel quale i record arrivano, vengono elaborati e fuoriescono nuovamente. Ai fini di un consumatore di informazione, può essere considerata una “*sorgente di record*”.
Il **TimeStamp** indica la coordinata temporale di creazione del record, in millisecondi a partire dal 01/01/1970. Applicativamente può aver significato solo il suo valore relativo ad altri record.
Il **SID** (Shell ID) indica l’area operativa del server da cui il record proviene.

Il **SeqNo** (Sequence Number) è molto importante nell'interscambio dei dati. Ciascun record applicativo viene etichettato con un numero progressivo nel momento in cui viene inoltrato dal server verso il client. Questo numero può quindi essere utilizzato per verificare che non ci siano record mancanti nella sequenza di arrivo. Inoltre se il flag di RESEND del canale è acceso, il server mantiene gli ultimi record inviati (fino a 10000) per permettere al client di richiedere il reinvio di una parte di essi se ci sono problemi di sequenza. Inoltre se il flag di RECOVERY è acceso, se la connessione cade, può essere ripristinata ed il client richiederà il reinvio utilizzando il SeqNo dell'ultimo record ricevuto correttamente.
**Size** indica quanti campi contiene il record: nella lettura si dovrà quindi procedere leggendo un egual numero di stringhe.
**Field 1..n** sono i campi del record (dati applicativi veri e propri). Il loro numero è specificato da 'size'.

## Compressione Delta

Se la compressione specificata per il canale è 'N', allora tutti i record saranno inviati esattamente con la struttura sopra indicata; ma se è specificata una compressione *delta* di profondità *n* (es. 'D1' profondità 1, 'D6' profondità 6, 'D12' profondità 12) il client autorizza il server ad inviare dei delta-record. Il server non è tenuto ad inviare delta-record, ma deciderà di farlo, record per record, se questo è vantaggioso in termini di banda utilizzata.
Un delta-record si distingue da un record normale dal flag DELTA acceso (bit 6). La lettura dei flag del record avviene quindi allo stesso modo sia per i record delta che per quelli normali: se il flag DELTA è spento, si procede alla lettura standard, mentre se è acceso si effettua una lettura speciale.
Tale lettura prevede che **sia il client che il server mantengano una lista (array) degli ultimi *n* record trasferiti**, dove *n* è la profondità della compressione. Un delta-record è un record che contiene solo le variazioni rispetto ad uno degli ultimi *n* record del buffer. Solo i campi differenti rispetto al record di riferimento vengono trasferiti, mentre gli altri potranno essere copiati dal record di riferimento.
Dopo la lettura dei flag e dopo aver rilevato che il flag di delta è acceso, se la profondità è maggiore di uno bisognerà leggere un intero che fornisce l'indice del record di riferimento nella lista. Se la profondità è 1 questo indice sarà ovviamente 0 e quindi non sarà trasferito.
Il TAGname sarà copiato direttamente dal record di riferimento, poiché un delta-record può avere come riferimento **solo** un record appartenente alla stessa TAG. Il TimeStamp, il SID ed il SeqNo invece dovranno sempre essere letti dalla socket. Il Size dovrà essere copiato dal record di riferimento, poiché un delta-record ha sempre la stessa dimensione del suo riferimento.
A questo punto non rimane che leggere i campi del record.
Essi vengono **suddivisi idealmente in gruppi di 32 campi** e per ciascun gruppo verrà spedita una mappa di bit (intero 32 bit) ciascuno dei quali specifica se il record corrispondente deve essere letto dalla socket o copiato dal record di riferimento.
Sono necessari quindi due contatori per i campi, uno globale, l'altro che si resetta a 0 dopo il 31esimo campo, dopo il 63esimo, il 95esimo, ecc. (in pratica pari a quello globale modulo 32).
Ogni volta che il contatore globale raggiunge un multiplo di 32 (compreso lo 0), si legge un intero (I) che costituisce la mappa dei bit per i prossimi 32 campi e si resetta a 0 il secondo contatore che sarà utilizzato per indicizzare i bit nella mappa.
Un bit a 0 indica che il campo deve essere letto dalla socket, a 1 indica che deve essere copiato dal campo corrispondente nel record di riferimento.
Ovviamente nella maggior parte dei casi l'ultima mappa di bit dovrà essere utilizzata parzialmente e i restanti bit ignorati, non essendo il numero dei campi del record un multiplo di 32.

Di seguito è riportato un semplice esempio in Java di lettura di record:

```
public Record(DataInputStream dis, Record[] last) throws IOException, EOFException
{
        int dimen;

        flags = dis.readInt();
        if ((flags & FL_DELTA) == FL_DELTA)  // LETTURA DELTA
        {
                int what = 0;
                if (last.length > 1)
                        what = dis.readInt();
                tagName = new String(last[what].tagName);
                ts = dis.readLong();
                sid = dis.readInt();
                seqno = dis.readInt();
                fieldValues = new String[last[what].fieldValues.length];
                int base=0;
                int offs=0;
```

```
                int dmap = 0;          // mappa delta
                for (int i=0; i<fieldValues.length; i++)
                {
                        if ((i & 31) == 0)
                        {
                                base = i;       // base
                                offs = 0;
                                dmap = dis.readInt();
                        }
                        if ((dmap & (1 << offs)) == 0)
                                fieldValues[i] = new String(last[what].fieldValues[i]);
                        else
                                fieldValues[i] = dis.readUTF();
                        offs++;
                }
        }
        else
        {
                tagName = dis.readUTF();       // LETTURA NORMALE
                ts = dis.readLong();
                sid = dis.readInt();
                seqno = dis.readInt();
                dimen = dis.readInt();
                fieldValues = new String[dimen];
                for (int i=0; i<dimen; i++)
                        fieldValues[i] = dis.readUTF();
        }
}
```

Rimane da descrivere la **gestione della lista** degli ultimi *n* record.
All'inizio (dopo la creazione del canale o dopo una recovery) la lista sarà vuota. All'arrivo di un record (delta o normale che sia) esso dovrà essere inserito in testa alla lista, facendo scrollare tutti gli altri eventuali record presenti di una posizione, se si verificano le due condizioni:

- È abilitata la compressione delta
- Il SeqNo del record arrivato è >= 0 (non negativo)

Infatti non tutti i record dovranno essere inseriti in questa lista: tipicamente i record di servizio (META record) hanno un SeqNo negativo e non devono essere utilizzati.

**Nota**: La compressione delta riguarda solo i record che viaggiano dal server al client: in nessun caso il client è autorizzato a spedire un delta-record. Questo significa che il client deve occuparsi solo della decodifica (lettura) e non della codifica (scrittura).

## I meta-record di ALIVE

I messaggi di ALIVE sono dei record di servizio utilizzati per riempire eventuali periodi di inattività del canale in modo da rassicurare le parti in comunicazione sul buono stato della stessa.
Un record di ALIVE è identificato da un TAGname particolare, "**META#ALIVE**" ed ha un unico campo, contenente un valore di verifica, da ignorare.
Tutti i record il cui TAGname inizia per "META#" sono detti meta-record: sono record di servizio e non devono essere, in genere, inoltrati allo strato applicativo.
Se il client richiede un canale con ALIVE = true, il server spedirà normalmente tutti gli altri record, ma se in questo invio si verificassero pause di oltre 100 secondi, tale lacuna verrà colmata con un record di ALIVE. Il client, ricevendo questo record può utilizzarlo o non utilizzarlo a suo piacimento.
Il client da parte sua è sempre libero di inviare o non inviare record di ALIVE, con la frequenza che ritiene opportuna, qualunque sia la specifica di canale, poiché il server non intraprende alcuna azione in base a questi record (vengono scartati in ogni caso).
Quindi il record di ALIVE può essere di utilità solo per il client.
Spedire record di ALIVE può servire al client per verificare lo stato della socket, in quanto se nessuno dei due comunica, potrebbe non accorgersi che la socket è caduta.
E' comunque consigliabile non spedire record di ALIVE troppo frequentemente e comunque non spedirli se altri record viaggiano nel canale.
*Nota*: I meta-record hanno, in genere, un numero sequenziale negativo, non incrementante (ad esempio sempre –1).

## I meta-record ACK e NAK

Se il canale ha il flag RESEND acceso, il server mantiene una lista degli ultimi record spediti verso il client (che non ha nulla a che fare con la lista per la compressione delta).

Il client riceve, da parte sua, una sequenza di record numerati da 0 (SeqNo) e, non considerando i record di servizio con SeqNo negativo, tale sequenza non può avere 'buchi'.
In condizioni di sequenza normale il client (che deve sempre verificare la sequenza) spedirà ogni *m* record ricevuti (con 10 < m <1000 consigliato 128) un record di ACK (acknowledged) indicante il SeqNo dell'ultimo record letto correttamente e coerente con la sequenza, in modo da permettere al server di eliminare devinitivamente quei record dal buffer di reinvio.
Il record di ACK è identificato da TAGname = "**META#ACK**" e possiede un unico campo contenente il SeqNo dell'ultimo record letto correttamente.
Se il client rileva un problema nella sequenza, come un record mancante o un record che arriva prima di un altro, dovrà spedire al più presto possibile un record di NAK (not acknowledged).
Il record di NAK è identificato da TAGname = "**META#NAK**" e possiede un unico campo contenente il SeqNo dell'ultimo record letto correttamente.
E' quindi strutturalmente identico al record di ACK, ma informa il server di compiere due azioni:

- Liberare dal buffer di resend tutti i record con SeqNo minore o uguale a quello indicato
- Rispedire tutto il buffer di resend residuo (cioè dal seqno indicato + 1 in poi)

Il client dovrà insistere a rispedire il NAK, finché non giungerà il record con il SeqNo atteso, allora potrà riprendere la normale sequenza.
A seconda del tipo di canale il client potrebbe dover ripetere il NAK al server diverse volte prima di essere ascoltato, per motivi di sincronizzazione fra le due direzioni del canale.

### Sottoscrizione di una TAG

Per ricevere uno o più flussi informativi (applicativi) il client dovrà sottoscrivere le rispettive TAG al server.
Tale azione consiste nel richiede una TAG tramite uno speciale meta-record e ricevere in risposta un meta-record contenente il *template* della TAG richiesta (tracciato record). Da quel momento in poi il client riceverà tutti i record di valore applicativo di quella TAG.
Il meta-record di sottoscrizione è identificato da TAGname = " **META#REQTAG**" ed ha 3 campi:

- Nome della TAG richiesta
- Filtro iniziale da applicare alla sottoscrizione
- Modalità di notifica iniziale da applicare alla sottoscrizione

I campi 2 e 3 saranno normalmente stringhe vuote, per ricevere tutta la TAG specificata
Ad esempio per richiedere la TAG "PROVA", si dovrà spedire un meta-record così composto

```
rec.tagName = "META#REQTAG";
rec.fieldValues[0] = "PROVA";
rec.fieldValues[1] = "";
rec.fieldValues[2] = "";
```

Questo meta-record viene interpretato ad un livello diverso degli altri e a differenza dei record ACK, NAK o ALIVE **deve avere un SeqNo valido** (positivo e in sequenza). Inoltre **deve avere il Flag FL_REQAN (request answer) acceso** poiché richiede un record di risposta che fornisca l'esito della sottoscrizione e il template della TAG.
*Nota*: Il client deve ricordarsi i SeqNo di tutti i record che invia e che richiedono risposta, poiché la risposta utilizzerà il SeqNo per indicare a quale record è relativa!
La risposta arriverà sotto forma di meta-record con TAGname = " **META#ANSWER#*xxx***" ove '*xxx*' è il SeqNo del record a cui la risposta si riferisce. Il meta-record ANSWER ha a sua volta SeqNo negativo.
Il contenuto dei campi del record di risposta dipende in tutto e per tutto da quale era la domanda: nel caso di una risposta al REQTAG **positiva** contiene 5 + *n**2 + *k* campi ove *n* è il numero dei campi del template, *k* è il numero dei campi chiave di accesso ai record.
I primi 5 campi forniscono

- Nome della TAG sottoscritta
- Livello di accesso alla TAG
- Area di pertinenza della TAG
- Numero dei campi
- Numero delle chiavi

Di seguito si trovano 2 campi per ogni campo del template, di cui il primo definisce il nome del campo e il secondo il tipo del dato di PSI.
I tipi di PSI indicano quale interpretazione applicativa dovrà essere fatta sui record e non ha nulla a che fare con i tipi base visti in precedenza. I tipi applicativi di PSI sono:

0 = stringa
1 = int16

2 = int32
3 = int64
4 = real (floating point)

seguono *k* campi indicanti gli indici delle *k* chiavi del record.
Esempio:

CLIENT spedisce:

```
rec.tagName = "META#REQTAG";
rec.SeqNo = 4;
rec.fieldValues[0] = “INDIRIZZI”;
rec.fieldValues[1] = “”;
rec.fieldValues[2] = “”;
```

SERVER risponde:

```
rec.tagName = "META#ANSWER#4";
rec.SeqNo = -1;

rec.fieldValues[0] = “INDIRIZZI”;
rec.fieldValues[1] = “1000”;
rec.fieldValues[2] = “1”;
rec.fieldValues[3] = “3”;
rec.fieldValues[4] = “1”;

rec.fieldValues[5] = “Nome”;
rec.fieldValues[6] = “0”;
rec.fieldValues[7] = “Via”;
rec.fieldValues[8] = “0”;
rec.fieldValues[9] = “Numero Civico”;
rec.fieldValues[10] = “2”;

rec.fieldValues[11] = “0”;
```

Il campo 3 dice che il template ha 3 campi (‘Nome’, ‘Via’ e ‘Numero Civico’), il campo 4 dice che uno solo di questi campi è un campo chiave. I campi 5 e 6 descrivono il primo campo del template (nome e tipo), così 7 e 8 descrivono il secondo e così via. Finiti i campi del template, il campo 11 specifica che il campo 0 del template (‘Nome’) è il campo chiave.
La risposta al REQTAG può però essere anche negativa, se, ad esempio, l’utente non ha sufficienti diritti di accesso per richiedere quella TAG.
In questo caso il META#ANSWER di risposta **negativa** sarà troncato al primo campo con un secondo campo **opzionale** che specifica il motivo del rifiuto.
Esempio:

```
rec.tagName = "META#ANSWER#4";
rec.SeqNo = -1;
rec.fieldValues[0] = “INDIRIZZI”;
rec.fieldValues[1] = “Insufficient clearing level”;    // campo opzionale: potrebbe non esserci
```

*Nota*: una risposta positiva ad una REQTAG ha sempre almeno 5 campi.
*Nota*: il numero dei campi di un record in lettura non deve mai essere assunto semanticamente: deve sempre essere considerato il Size del record.
*Nota*: dopo una risposta positiva ad una REQTAG dovrebbero seguire i record dati relativi alla TAG richiesta, tuttavia il client non deve mai assumere che possa esserci sequenzialità fra le due cose, anche se in pratica accade. In altre parole il client deve essere pronto a memorizzare record dati subito dopo la richiesta fatta, non subito dopo la risposta ricevuta!

### Gli info-record

Gli info-record sono una variante dei meta-record che non forniscono informazioni operative sul canale, ma informazioni generiche che il client deve semplicemente memorizzare e utilizzare se se ne presenta l’occasione.
Il record “**INFO#MYNAME**” comunica al client il nome del gate di servizio del server periferico.
Tale record ha un singolo campo che contiene appunto tale nome.

Il nome del gate di servizio è utile al client nel caso di connessioni dirette tramite chiavi locali, per permettere l'operazione di recovery.

## La RECOVERY

Se un record arriva non in sequenza, come già visto, il client potrà richiedere il reinvio di alcuni record tramite il meta NAK: a questo livello l'interscambio dei record è ancora efficiente e non è necessaria una recovery.
Ma se ci sono problemi di scrittura/lettura sulla socket, oppure il client perde il sincronismo sui dati e comincia a ricevere informazioni incoerenti (dalla verifica dei tipi base), allora potrà effettuare una recovery del canale se sul canale è stata richiesta la recovery (flag relativo abilitato) prima di giungere alla drastica decisione di creare un altro canale e ricominciare tutto daccapo.
Effettuare una **recovery evita di dover risottoscrivere daccapo tutte le TAG** e di riceve nuovamente alcuni dati già trasferiti (questo secondo aspetto dipende dalla natura dei dati).
Per effettuare una recovery il client deve conoscere il nome del gate di servizio con cui stava comunicando in precedenza e richiedere al server di fornirgli lo stesso gate.
Per fare questo il client aprirà una nuova socket, ma utilizzerà come header HTTP l'header di recovery

**RECOVERY /nomegate -7854972\r\n\r\n**

ove 'nomegate' è il nome del gate che il client può aver ricevuto tramite il record "**INFO#MYNAME**"
Se il client non ha ricevuto questo record, non potrà effettuare la recovery e dovrà procedere alla apertura di un canale su un nuovo gate (procedura standard di apertura della comunicazione: CREATE).
Dopo l'header di recovery si dovrà comunque procedere alle operazioni di **verifica utente** e alla comunicazione delle **specifiche di canale** che dovranno essere identiche a quelle utilizzate nella CREATE
A questo punto il client riceverà il flusso normale (quello che aveva perso) senza dover risottoscrivere le TAG, ma sicuramente alcuni sequenziali saranno andati persi, quindi il client dovrà utilizzare il meta-record **NAK** per recuperarli, come descritto in precedenza.
*Nota*: un canale può essere ricoverato solo entro un minuto dal primo errore sull'IO. Dopo tale periodo il server elimina il gate di servizio e la recovery non sarà più possibile. E' quindi necessario che il client rilevi tempestivamente i problemi di IO sulla socket o di sincronismo sui dati.
*Nota*: Per utilizzare la recovery è necessario accendere sia il bit di RECOVERY che quello di RESEND, altrimenti il NAK non funzionerà e, nonostante la recovery si saranno persi dei dati.