Interoperabilità in Sistemi Informativi Sanitari realizzata attraverso piattaforma Mirth Connect

Gregorio Canal
Matricola: 1079962
Relatore: *Prof. Giovanni Sparacino*
Correlatore: *Ing. Cristiano Rizzo*, Micromed S.p.A., Mogliano (TV)
Università degli Studi di Padova
Facoltà d'Ingegneria
Laurea Magistrale in Bioingegneria

6 Ottobre 2015

Indice

Capitolo 1

Interoperabilità nei Sistemi Informativi Sanitari e caso di studio

1.1 Introduzione

Da circa 20 anni a questa parte si è verificata una rivoluzione nella Sanità. Le tecnologie informatiche, che prima venivano utilizzate solamente per automatizzare i processi amministrativi, hanno cominciato ad essere impiegate in attività relative al processo di cura del paziente quali:

- il miglioramento del processo riabilitativo
- il miglioramento del processo di cura dei pazienti in fase acuta
- il miglioramento dell'assistenza ai malati cronici
- la facilitazione della prevenzione di malattie
- l'aumento dell'efficacia e dell'efficienza del lavoro quotidiano degli operatori sanitari.

In pratica l'informatica sta diventando, sempre più, uno strumento clinico da utilizzare in Sanità ed uno strumento di ricerca per la Medicina. In particolare, alcune esigenze nell'ambito medico sanitario, viste in *Sistemi Informativi d'Impresa* di Bracchi Giampio etc.[2], di cui l'Informatica Medica si occupa sono:

1. riduzione della distanza fra tecnologia e medicina: affinchè il lato medico sappia utilizzare gli strumenti proposti e allo stesso tempo formare chi propone le soluzioni tecnologiche ad avere una buona conoscenza delle problematiche da dover affrontare
2. razionalizzazione dei processi sanitari: agli inizi degli anni '90 è stata fatta una riforma del Servizio Sanitario Nazionale (SSN) nella quale sono stati introdotti tre nuovi ed importanti concetti, si veda *Sistemi Informativi per la Sanità* di Teti Antonio e Festa Giuseppe[10]:
    - La *Regionalizzazione*, che ha l'obiettivo di affidare alle singole Regioni il compito di organizzare e strutturare il proprio Servizio Sanitario Regionale (SSR) in termini di tutela della salute, criteri di finanziamento e

attivitá di controllo sulla gestione e sulla qualità dei servizi erogati, mantenendo, tuttavia, un livello assistenziale uniforme su tutto il territorio nazionale.

- L'*Aziendalizzazione*, con cui si intende il processo di trasformazione delle precedenti Unità Sanitarie Locali (USL) in Aziende Sanitarie Locali (ASL) e degli ospedali principali in Aziende Ospedaliere (AO). Le AO e gli istituti privati vengono quindi retribuiti dalle ASL in base alle prestazioni erogate
- Il *Finanziamento a prestazione*, con il quale sono stati stabiliti due criteri di finanziamento con cui le Regioni retribuiscono le ASL:
  - La quota capitaria: cioè un importo fisso basato sul numero di residenti nel territorio della singola ASL
  - La retribuzione per prestazioni erogate: cioè un importo variabile in funzione dei ricoveri, interventi, esami, etc. erogati

3. Metodologie efficaci, efficenti e sicure di gestione dati: ciò significa ricercare un modo con cui i dati dello specifico paziente possano essere organizzati e trattati in maniera razionale, efficiente e sicura. Uno degli argomenti piú discussi in questo ambito è la cartella clinica elettronica

4. Metodologie di interpretazione dei dati a fini scientifici o clinici: riguarda, per esempio, l'identificazione di algoritmi o metodologie che possano dare supporto al medico nella decisione in ambito clinico

Si può quindi dire che l'Informatica Medica ha come scopo il miglioramento, in maniera diretta o indiretta, del processo di cura del paziente, ed i mezzi di cui si serve per raggiungere tale scopo sono:

- algoritmi per l'analisi dei dati e la loro elaborazione
- sistemi per la gestione dei dati clinici in maniera consistente, efficiente e sicura
- sistemi di trasmissione/comunicazione/esportazione di dati clinici basati su metodologie informatiche e telecomunicazioni
- sistemi di supporto alla decisione clinica.

L'applicazione dei principi della Teoria dell'Informazione alla Sanità ha dato luogo all'informatizzazione delle realtá ospedaliere e con queste alla definizione dei Sistemi Informativi Sanitari.

1.2 Sistemi Informativi Sanitari

Il processo di cura di una persona produce una quantità notevole di dati clinici a causa delle molteplici entità coinvolte nel processo, quali le attività gestionali dei reparti in cui il paziente transita, la strumentazione utilizzata, gli esami effettuati. In figura 1.1 é possibile vedere i flussi informativi generati dal processo di cura di un paziente dall'arrivo in pronto soccorso alla dimissione. Durante il ricovero l'ospedale effettua, per conto del paziente, delle richieste di prestazione ai vari reparti che poi provvederanno ad erogare il servizio richiesto. Se per esempio viene inviata la

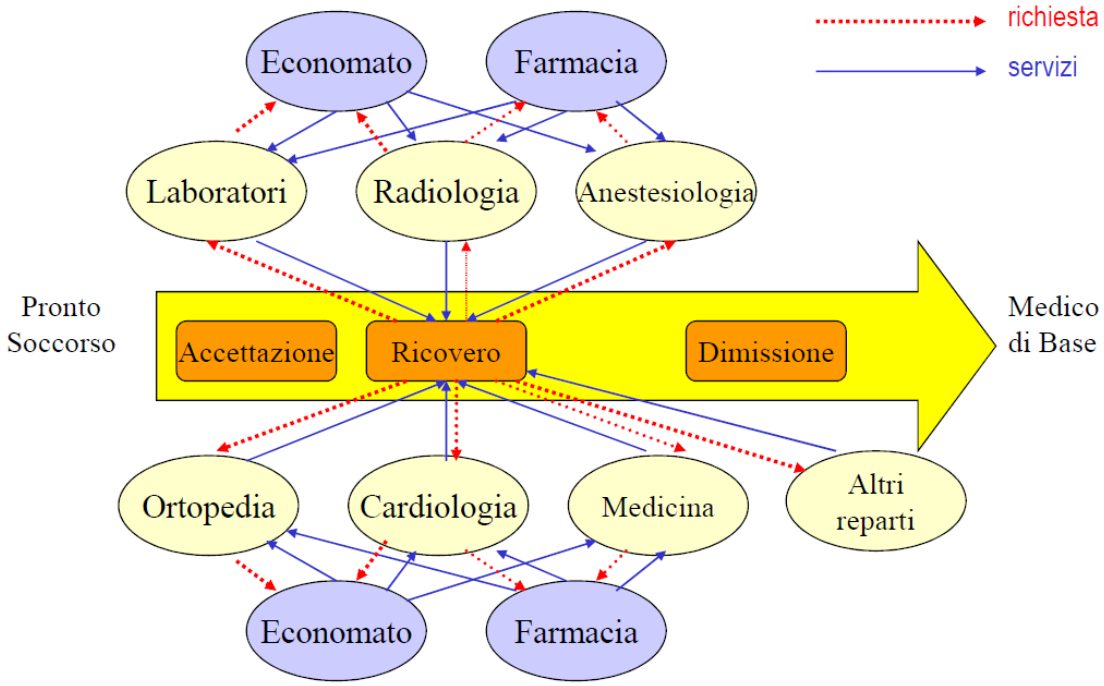

Figura 1.1: Diagramma del flusso informativo generato nel corso della degenza del paziente

richiesta di effettuare un EEG al reparto di Neurofisiologia, una volta erogato il servizio, il risultato dell'esame viene messo a disposizione del medico curante.
Allora, la finalità di un SIS è la gestione di informazioni utili alla misura ed alla valutazione dei processi gestionali e clinici al fine di ottimizzare le risorse impiegate nel conseguimento degli obiettivi istituzionali e ottimizzare le modalità di comunicazione. I SIS hanno, dunque, il compito di:

- supportare la pianificazione ed il controllo a livello nazionale e regionale
- agevolare l'assistenza nel rapporto tra cittadino e medico
- migliorare l'erogazione delle prestazioni

In Italia, per adempiere a questi compiti, i SIS si sono sviluppati in due filoni principali:

1. l'informatizzazione dei flussi di controllo nazionale e regionale
2. l'informatizzazione dell'erogazione delle prestazioni nelle aziende sanitarie.

I Sistemi Informativi Sanitari presenti sul nostro territorio si possono distinguere in tre livelli:

1. Nazionale: si occupa dei flussi informativi statistici e di controllo sulle attività sanitarie, per esempio il monitoraggio delle informazioni legate alla natalitá, del controllo delle principali aree di spesa, quali le prescrizioni farmaceutiche, e del coordinamento delle azioni regionali sul territorio.
2. Regionale: si occupa di fornire supporto alla pianificazione ed al controllo dell'assistenza, ai medici di medicina generale, ai cittadini. L'elemento chiave di questo sistema è il tesserino sanitario cha abilita l'autenticazione dell'assistito, l'accesso ai servizi sanitari in rete, la consultazione delle proprie informazioni sanitarie via internet.

3. Territoriale o Locale: tale sistema è imperniato sulla ASL e produce servizi amministrativi ed anagrafici, assitenziali e di prevenzione. Tali servizi sono erogati in modo integrato dalle strutture ASL e dai medici di base e aggiornano la storia clinica di ogni paziente alimentando così i flussi informativi economo-finanziari verso la Regione ed il Ministero.

4. Ospedaliero: tale sistema è quello più a stretto contatto con il paziente ospedalizzato ed eroga servizi sanitari specialistici, gestisce informazioni complesse, quali la cartella clinica, e può essere geograficamente distribuito. Inoltre, deve gestire dei processi primari, quali l'accettazione, l'inquadramento clinico, la diagnosi, la cura e la dimissione, e dei processi di supporto, quali attività strategiche ed infrastrutturali, la gestione delle tecnologie, l'approvvigionamento e la logistica.

Il filone 2. applicato al SIS Ospedaliero verrà approfondito successivamente perchè in stretto legame con l'obiettivo della tesi. Per sopperire, dunque, all'informatizzazione dell'erogazione delle prestazioni, le Aziende Ospedaliere nel territorio sono spesso strutturate in tre settori ognuno dei quali ha una diversa area di applicazione:

- **Amministrativa e Direzionale:** tale area include sistemi che supportano i processi amministrativi, la gestione delle risorse e i processi di pianificazione e controllo.

- **Clinico Sanitaria:** tale area riceve le richieste dal CUP o dal sistema di gestione ordini e, conseguentemente alle richieste, dà supporto all'esecuzione degli esami. Una volta raccolti i risultati, sotto forma di annotazioni o direttamente dalle strumentazioni, comunica al richiedente l'avvenuta esecuzione dell'esame e ne fornisce il referto.

- **Front Office:** tale area riguarda il decorso della degenza del paziente nella struttura ospedaliera ed è suddivisibile in tre parti:

  - *Accettazione-Dimissione-Trasferimento:* L'ADT è una funzione che generalmente è centralizzata nel Sistema Informativo Ospedaliero che serve i diversi reparti e dà loro supporto nella gestione amministrativa dei pazienti. Per esempio, tiene traccia della movimentazione dei pazienti, comunica ai reparti l'accettazione di un paziente e registra l'esito del ricovero ricevuto dagli stessi.
  - *Centro Unico Prenotazioni:* il CUP è una componente unica e centralizzata del SIS Ospedaliero che serve le unità erogatrici dell'azienda quali ambulatori, laboratori, etc: si occupa delle agende di prenotazione degli ambulatori, registra l'avvenuta erogazione delle prestazioni, ecc.
  - *Pronto Soccorso:* supporta la registrazione di nuovi pazienti, la dimissione ed il trasferimento degli stessi verso le strutture aziendali di ricovero.

L'attività clinica nei processi di accettazione e degenza di un paziente è caratterizzata dalla molteplicità degli attori, dalla multidisciplinarietà e dal flusso informativo elevato. Queste caratteristiche fanno emergere la necessità di poter condividere le infromazioni che sono frammentate fra medici, infermieri, specialisti, medico di base e familiari in maniera immediata e chiara per tutte le persone che ne devono venire in contatto. In quest'ottica le strutture sanitarie si stanno orientando alla

digitalizzazione dei dati anagrafici, all'interoperabilità tra i sistemi, per ottenere un riconoscimento unico del paziente, e all'informatizzazione della cartella clinica, per poter accedere agli esami fatti nel corso della degenza ed alla intera storia clinica. Poichè l'accesso all'intera storia clinica di un paziente ha insita in sè l'eterogeneità degli attori che hanno concorso al suo sviluppo, occorrono degli standard di integrazione che assicurino l'interoperabilitá delle informazioni scambiate.

1.3 Livelli di Standardizzazione nei SIS

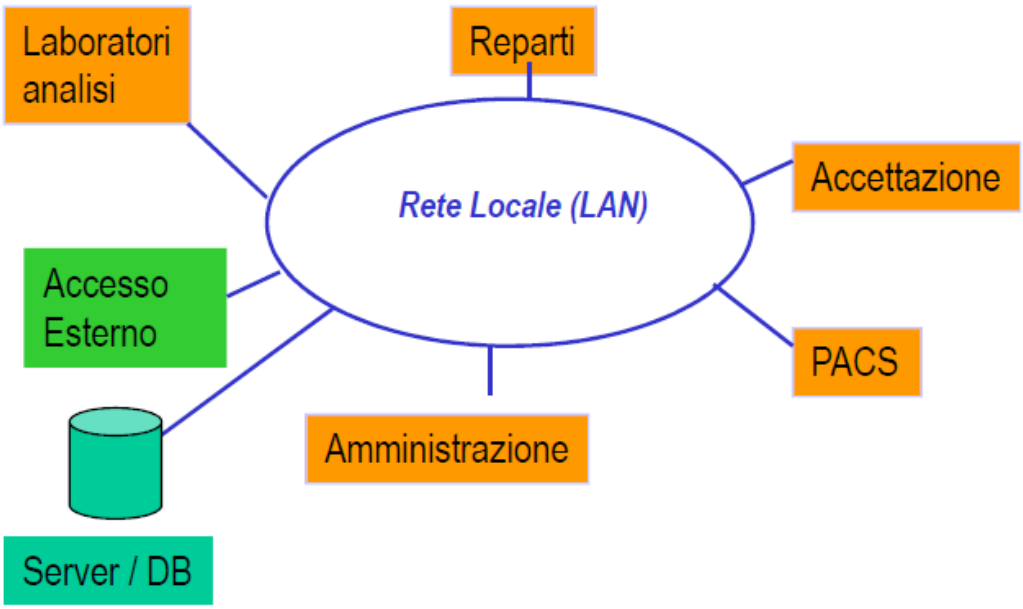

Figura 1.2: Esempio di architettura distribuita, o Client-Server, per un SIS

L'Architettura generale di un Sistema Informativo Sanitario può avere, come possiamo vedere in figura 1.2, una configurazione di tipo **Client-Server**. Poichè in un SIS sussistono molti servizi, ognuno dei quali ha delle esigenze operative diverse, l'architettura spesso più utilizzata é quella distribuita, in cui le procedure e le applicazioni rimangono specifiche dei vari servizi, distribuiti su diverse macchine che vengono messe in comunicazione fra loro attraverso una connessione LAN. In questo modo, ciò che viene messo in comune sono solo le informazioni relative ai processi e alle applicazioni dei singoli servizi. Tale architetttura è, ovviamente, più flessibile rispetto a quella monolitica, meno costosa, poichè non è necessaria la progettazione di un' unica macchina multi-utente e multi-scopo, ma ogni servizio adotta il suo software. Un' architettura di questo tipo solitamente è costituita da:

- Un *Server* o *Database* centrale
- Vari *poli dipartimentali*, ognuno con le proprie applicazioni e processi, per esempio i laboratori, l'amministrazione, etc.
- *Poli di servizio*, ai quali si possono connettere i medici o i pazienti per il ritiro dei referti
- Una *rete LAN* che mette in comunicazione fra loro le componenti del SIS

La soluzione Client-Server è la più utilizzata sia per motivi economici che di flessibilità, ma tale architettura mette in evidenza il fatto che ciò che viene condiviso fra i vari servizi sono le informazioni. È quindi necessaria l'integrazione delle diverse applicazioni in maniera tale che le informazioni risultino chiare e comprensibili alle applicazioni ed agli operatori che dovranno farne uso.
Le informazioni all'interno di un SIS possono essere strutturate in maniera:

- **Informale:** i dati vengono salvati mediante l'utilizzo di una struttura minima, che è preferibile nel caso di informazioni relative a processi infrequenti o ad alta variabilità o non prevedibili.

- **Formale:** i dati sono strutturati in base ad un modello ben definito e dal quale non si può uscire. Dati strutturati in questa maniera sono ordinati, affidabili, efficienti e consentono una più agevole gestione dei dati, tuttavia è una soluzione più costosa e rigida e non molto amata dagli operatori che devono inserire i dati.

All'interno di un SIS sono presenti dati sia strutturati in maniera formale che informale in base ad una analisi costo-beneficio effettuata dagli enti dirigenti del SIS. Anche questa situazione, di diversità di strutturazione dell'informazione, ci riporta ad un problema di integrazione delle varie componenti di un SIS per poter scambiare in maniera semplice ed efficace le informazioni.

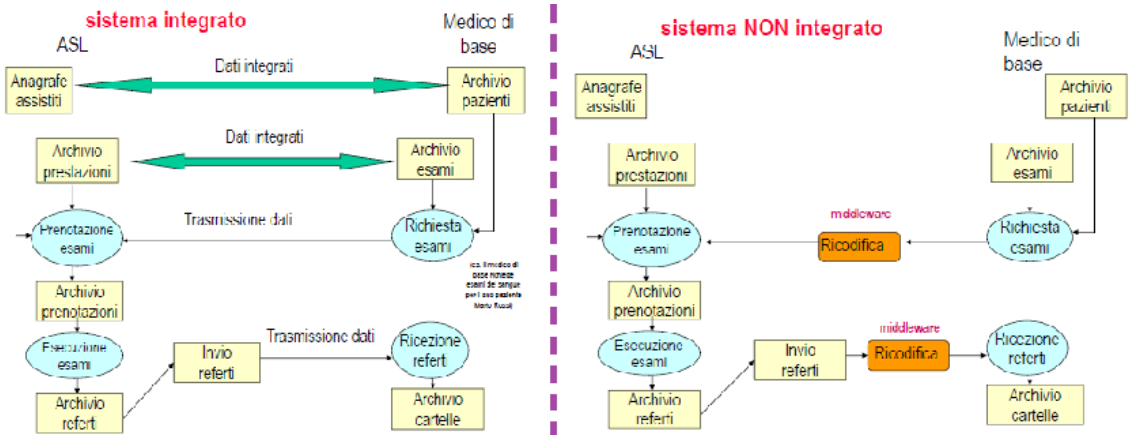

Figura 1.3: Diagrammi di flusso informativi di un SIS: a sinistra nel caso integrato a destra non integrato

Se nella progettazione iniziale di un SIS è presente un modello di dati, che sia comune ai vari sistemi, si può arrivare ad una soluzione integrata, in cui le informazioni scambiate fra i vari sistemi sono interpretabili da ogni sistema presente nel SIS.
Se non è presente un modello comune per tutte le componenti, il SIS risulta non integrato, ma con l'ausilio di alcuni software specifici, detti *middleware*, si può assicurare in ogni caso l'interoperabilità fra le varie componenti del SIS. I middleware possono, ad esempio, trasformare le informazioni secondo certi modelli in modo che le informazioni scambiate fra due componenti del SIS, anche se generate secondo modelli strutturali diversi, risultino comprensibili.
Pensare ad un Sistema Informativo Sanitario completamente integrato è, dunque, irrealistico in quanto non esiste, ancora, un modello di dati che possa essere comune per tutti oltre al fatto che in un SIS ogni reparto adotta dei Software che gestiscono i dati in maniera diversa l'uno dall'altro. Ma anche se la completa integrazione al

momento risulta irrealistica, è possibile pensare a Sistemi Informativi *parzialmente integrati*. Anzi la realtá nel panorama ospedaliero mondiale è per la maggior parte questa: Sistemi Informativi Ospedalieri costituiti da diverse piattaforme, relative ai vari reparti, con caratteristiche hardware e software sempre differenti ma che possono scambiare correttamente informazioni fra di loro grazie all'inserimento di un software di intermezzo che ne possa assicurare l'interoperabilità. Tali software sono detti *middleware* proprio perchè vengono interposti fra i due sistemi che si devono scambiare informazioni. In pratica sono dei postini che però non si limitano a consegnare le informazioni ma le trasformano in modo che il sistema destinatario possa correttamente comprenderle ed utilizzarle.
Gli attori presenti nel SIS hanno in dotazione software diversi, ed ognuno di questi organizza i dati a modo suo: per poter assicurare l'interoperabilità fra questi sistemi è dunque necessario standardizzare il SIS secondo tre aspetti:

1. **La semantica:** standardizzazione semantica significa che le informazioni di uno stesso tipo debbano essere strutturate allo stesso modo (che abbiano cioè lo stesso grado di formalità),

2. **La terminologia:** per standardizzazione terminologica si intende l'utilizzo di un sistema di nomenclatura e codifica, per i vari termini utilizzati all'interno di un SIS, che sia trasversale a tutte le componenti del SIS. In pratica, l'utilizzo di un solo codice per tutti i possibili sinonimi dello stesso termine. Lo standard utilizzato è l'ICD-9-CM

3. **La sintassi:** per standardizzazione sintattica si intende l'utilizzo di un linguaggio che sia comune a tutti i dispositivi che si devono scambiare informazioni all'interno di un SIS. Esistono vari linguaggi standard per l'interoperabilità tra i sistemi, ma quello più utilizzato ed in rapida diffusione, anche in Italia, è l'Health Level Seven-HL7.

1.4 Lo Standard HL7

Fondata nel 1987, la Health Level Seven International, come ampliamente spiegato nel sito `www.hl7.org`[2], è un'organizzazione accreditata dall'ANSI per lo sviluppo di standard per lo scambio e la condivisione di informazioni sanitarie, a livello elettronico, che possano dare supporto clinico e valutare e gestire i servizi sanitari.
L'HL7 non è un linguaggio di programmazione, è piuttosto un insieme di regole per agevolare la comunicazione fra applicazione ed applicazione. Infatti, il numero 7 si riferisce proprio al livello di Applicazione dello standard ISO/OSI per i protocolli di comunicazione.
Poichè l'HL7 è stato realizzato per lo scambio di informazioni cliniche fra dispositivi all'interno di un SIS, che utilizzano piattaforme hardware e software diverse, esso deve far fronte a diverse tipologie di dati che vengono condivisi. Per far fronte a questo problema l'HL7 mette a disposizione diversi tipologie di messaggi, ognuna delle quali è relativa ad un certo evento.
Vediamo di seguito quali sono i più utilizzati:

- *ACK:* utilizzato per la gestione di messaggi di Acknowledgment

- *ADT:* utilizzato per la gestione di informazioni relative all'Accettazione, alla Dimissione ed al Trasferimento dei pazienti

- *BAR:* utilizzato per la gestione di informazioni relative all'aggiunta o alla modifica delle fatture
- *DFT:* utilizzato per la gestione di informazioni relative a transazioni finanziarie dettagliate
- *MDM:* utilizzato per la gestione di informazioni che riguardano la gestione di documenti medici, quali i referti
- *ORM:* utilizzato per la gestione di messaggi che riguardano richieste d'esame
- *RAS:* utilizzato per la gestione di messaggi che riguardano il trattamento del paziente

I messaggi di tipo ORM, ADT, ACK e MDM sono quelli che verranno più utilizzati perchè riguardanti lo scopo della tesi.
I messaggi HL7 sono composti in testo ASCII ed hanno una struttura gerarchica molto precisa:

- ogni messaggio è suddiviso in più **segmenti** ognuno dei quali riporta delle informazioni relative ad uno specifico ambito della richiesta contenuta del messaggio. Ad esempio, il messaggio in figura 1.4 é composto da 5 segmenti :
  - MSH: il Message Header, in cui vengono inserite tutte le informazioni relative al tipo di messaggio (ORM,MDM, etc.), all'applicazione che ha inviato il messaggio e a quella che lo deve ricevere, etc.
  - PID: il Patient Identification, che include informazioni relative all'identificazione del paziente, quali nome, cognome, sesso, indirizzo, etc.
  - PV1: il Patient Visit information, che include informazioni ambulatoriali o relative al ricovero di un paziente, quali il posto letto assegnato, il nome e l'identificativo del dottore che ha richiesto il ricovero o la visita.
  - OBR: l'Observation Request, che include i dettagli relativi ad esami o studi diagnostici o alla valutazione dei risultati ottenuti da un esame.
  - ORC: il Common Order, che contiene le informazioni relative alla richiesta che si sta effettuando con la consegna del messaggio, per esempio il tipo di esame che bisogna effettuare, il dottore richiedente etc.

- ogni segmento è suddiviso in più **campi**, in questi vengono inserite le informazioni relative al segmento a cui il campo fa riferimento. Ad esempio il segmento MSH in figura 1.5 contiene 19 campi, ognuno dei quali deve contenere una specifica informazione. Di seguito vediamo, in ordine, alcuni dei campi che sono contenti nel segmento MSH:
  1. *Field separator:* deve riportare il carattere speciale che identifica in che modo nel messaggio si possono distinguere i campi. Solitamente tale carattere è predefinito di default ed è |.
  2. *Encoding Character:* caratteri speciali utilizzati per identificare la struttura del messaggio. Anche questi caratteri sono definiti di default e sono: $\widehat{\ }\sim$ \&, che identificano rispettivamente il separatore di componenti, il separatore di ripetizioni, il carattere d'uscita ed il separatore di sottocomponenti.

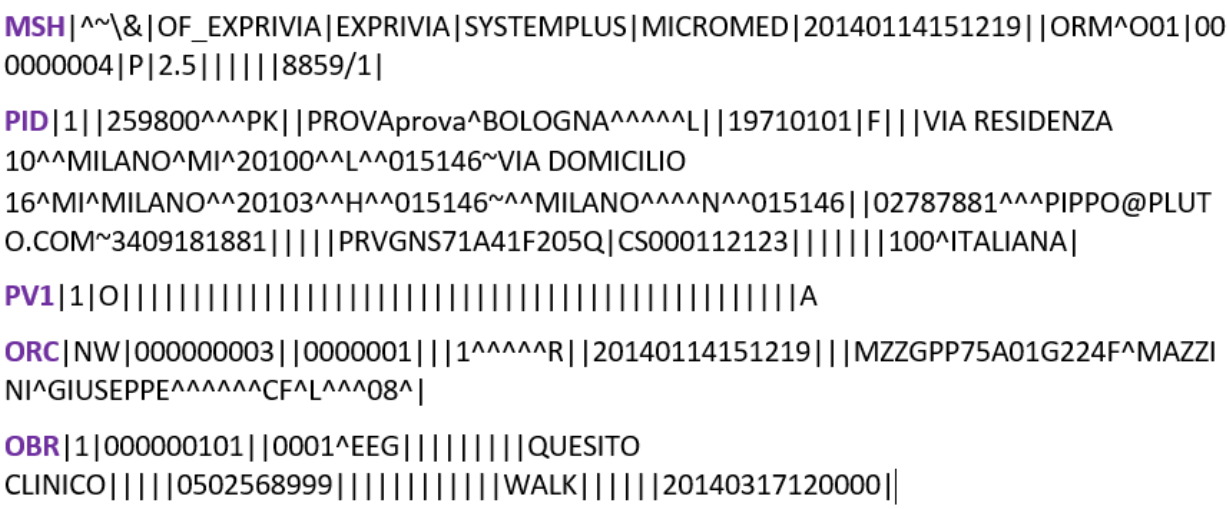

Figura 1.4: Tipologia di mesaggio HL7

3. *Sending Application:* campo che riporta il nome dell'applicazione che ha generato il messaggio.
4. *Sending Facility:* campo che riporta il nome della strutture mittente del messaggio generato dall'applicazione
5. *Receiving Application:* campo che riporta il nome dell'applicazione che deve ricevere il messaggio
6. *Receiving Facility:* campo che riporta il nome della strutture destinataria del messaggio generato dall'applicazione
7. *Date/time of message:* data e ora di generazione del messaggio
8. *Message Type:* tipo del messaggio

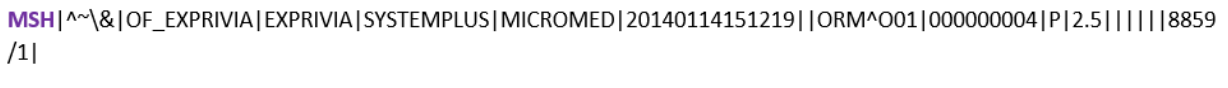

Figura 1.5: Segmento MSH di un messaggio HL7

- ogni campo può essere suddiviso in più **componenti**, in questo modo si possono arricchire le informazioni contenute nel campo a cui la componente fa riferimento. Per esempio nel segmento MSH il campo *Message Type* è suddiviso in due componenti: la prima riporta il tipo di messaggio, la seconda riporta l'evento che ha dato luogo al messaggio. Ad esempio in figura 1.5 il campo *Message Type* è valorizzato con ORM^O01, che significa che il tipo del messaggio è una richiesta esame (ORM) generata a motivo dell'evento O01.

1.5 Un caso di studio nell'azienda Micromed S.p.A.

Questo elaborato è nato dalla collaborazione con Micromed S.p.A, sita in Via Giotto 2, Mogliano Veneto (TV). L'azienda produce, sia a livello Hardware che Software, dispositivi elettromedicali indirizzati all'indagine Neurofisiologica. In particolare, i prodotti disponibili sono apparecchiature per Elettroencefalografia Digitale

(EEG, Video EEG, Stereo EEG), Holter EEG, Potenziali Evocati, Elettromiografia e Polisonnografia.
In particolare Micromed sviluppa e mette a disposizione dei clienti System Plus Evolution, un software che consente la gestione dei prodotti acquistati e la generazione dei referti degli esami effettuati.
System Plus Evolution è un software che, una volta acquistato, necessita di essere integrato nel Sistema Informativo Sanitario Ospedaliero, o, più brevemente, Sistema Informativo Ospedaliero (SIO). Per integrazione si intende la corretta interazione fra due diverse piattaforme software, nel caso specifico la corretta interazione fra System Plus Evolution e le diverse componenti del Sistema Informativo Ospedaliero.
Per l'integrazione di System Plus Evolution in un SIS, Micromed ha sviluppato un programma: la *Micromed HIS Interface*, il cui scopo è adempiere allo scambio di semplici informazioni con il Sistema Informativo Ospedaliero(in inglese Hospital Information System - HIS), del tipo :

- richiesta di esecuzione di un esame in System Plus Evolution (SIO → System Plus Evolution)
- invio dell'Acknowledgment per l'avvenuta ricezione della richiesta (System Plus Evolution → SIO)
- invio dell'Acknowledgment per l'avvenuta esecuzione dell'esame (System Plus Evolution → SIO)
- cancellazione/modifica della richiesta di un esame (SIO → System Plus Evolution)
- inserimento di un nuovo paziente nel Database (SIO → System Plus Evolution)
- invio del referto dell'esame eseguito (System Plus Evolution → SIO)
- aggiornamento dei dati del paziente (SIO → System Plus Evolution)
- fusione di due pazienti (SIO → System Plus Evolution)
- cancellazione di un paziente (SIO → System Plus Evolution)

In figura 1.6 si può vedere il modo in cui viene gestito attualmente il flusso di informazioni tramite l'utilizzo della Micromed HIS Interface. Si noti come il flusso informativo fra HIS e System Plus Evolution puó essere essenzialmente suddiviso in due parti:

- **Comunicazione tra HIS e Micromed HIS Interface:** le informazioni vengono scambiate tramite protocolli TCP/IP in linguaggio HL7, uno standard definito per lo scambio di informazioni medico sanitarie.
- **Comunicazione tra Micromed HIS Interface e System Plus Evolution:** una volta ricevute dalla Micromed HIS Interface, le informazioni vengono tradotte e salvate nell'*Exchange Database* da cui successivamente vengono importate in System Plus Evolution per effettuare l'esame richiesto.

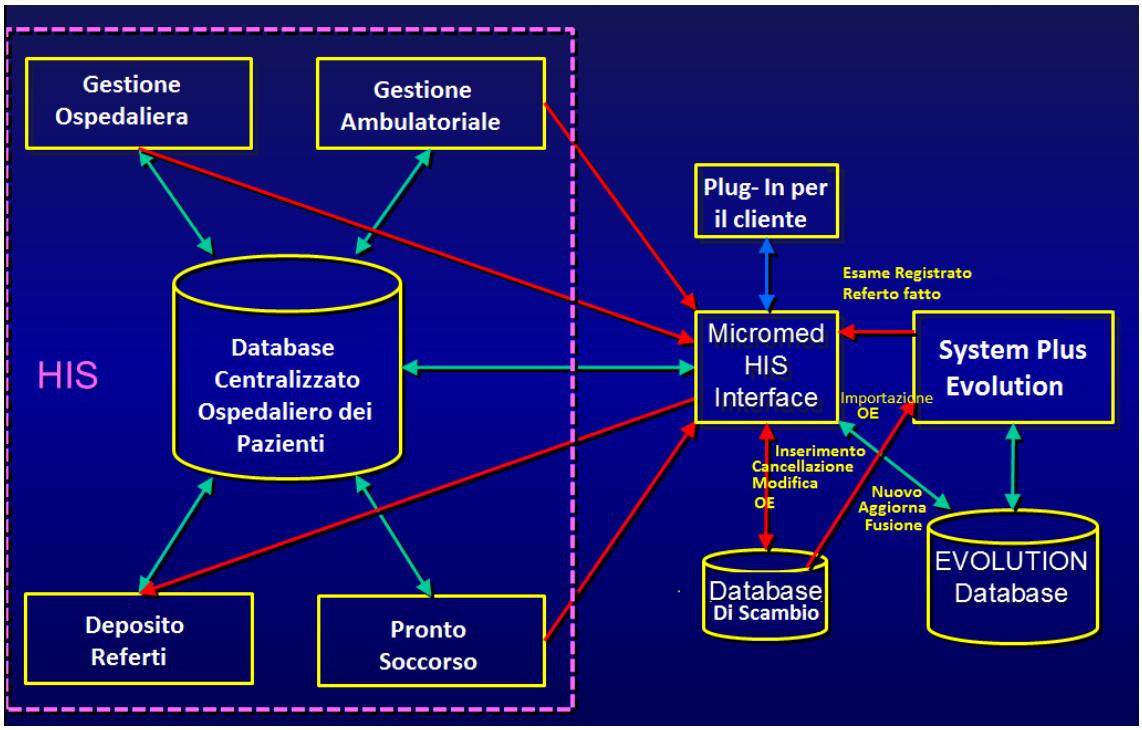

Figura 1.6: Diagramma del flusso informativo fra HIS e System Plus Evolution

1.6 Scopo della tesi

La standardizzazione HL7 avviene, tuttavia, solo a livello sintattico e non semantico, perciò ogni azienda che produce software con cui vengono integrate, nativamente, le diverse componenti di un SIS, ha un certo grado di libertà con cui può generare la messaggistica che viene utilizzata. Nelle sezioni precedenti è già stato approfondito cosa ciò significa, in ogni caso basti pensare metaforicamente che gli ospedali parlano tutti la stessa lingua, l'HL7, ma non è stato stabilito cosa si debbano dire.
Micromed, vendendo i suoi prodotti in tutto il mondo, ha fatto esperienza del problema dell'interoperabilità fra il proprio software e il SIO dei suoi clienti. Perchè la Micromed HIS Interface, utilizzata per comunicare con il SIO, andava adattata di volta in volta alla situazione dello specifico ospedale, richiedendo quindi ingenti investimenti di risorse. Inoltre alcune problematiche riscontrate in certi ospedali non erano risolvibili semplicemente adattando la Micromed HIS Interface. Per esempio al Sant'Orsola di Bologna balzava all'occhio la problematica nell'invio del referto: in pratica, non era stato previsto un meccanismo, da parte della Micromed HIS Interface, di reinvio del referto se per qualche motivo la connessione cadeva e, di conseguenza, a volte, andava perduto; o ancora, al San Raffaele di Milano la Micromed HIS Interface non riusciva sempre a tenere il passo con le molteplici richieste di connessione che arrivano dai vari reparti, causando, a volte, il blocco della Micromed HIS Interface. Altre necessitá dell'azienda che in generale era necessario risolvere sono:

- la necessità di trasformare i messaggi in maniera che siano comprensibili ed utilizzabili da chi ne è il destinatario, quindi sia dal lato Micromed che dal lato ospedaliero
- creazione di un file Log che tenga traccia dei messaggi in arrivo ed in uscita alla Micromed HIS Interface ed i relativi messaggi di acknowledgment

- filtraggio dei messaggi qualora non fosse giá attivo un filtro sui messaggi che arrivano alla Micromed HIS Interface

In questo scenario si inserisce la scopo della presente tesi: la valutazione di un middleware da interporre fra l'HIS e la Micromed HIS Interface. Una valutazione fatta nell'ottica della risoluzione dei problemi che si verificavano nelle singole realtà ospedaliere e della standardizzazione della Micromed HIS Interface, in maniera, quindi, da dover, in futuro, modificare, in base alle esigenze dell'ospedale, solo i parametri del middleware e non della Micromed HIS Interface. Il middleware che ho dovuto valutare è Mirth Connect, un programma open source che permette la trasformazione, il filtraggio ed il routing dei messaggi tramite diversi protocolli fra le varie componenti del SIS.
Il lavoro svolto nell'azienda si può essenzialmente dividere in due fasi: la comprensione e valutazione di Mirth e la messa in opera di due interfacce. Per quanto riguarda la comprensione e la valutazione sono stati fatti degli studi preliminari sul programma per capirne le funzionalità e comprenderne, almeno in parte, le modalità di utilizzo in modo da capire se effettivamente l'utilizzo di Mirth come middleware potesse risolvere le problematiche riscontrate, tali studi sono riportati nel capitolo 3 e 4 . Una volta riscontrato che Mirth Connect potesse, sulla carta, risolvere le problematiche degli Ospedali si è passati all'implementazione negli ospedali: prima al Sant'Orsola di Bologna poi al San Raffaele di Milano. I risultati ottenuti nella seconda fase sono riportati nel capitolo 4.

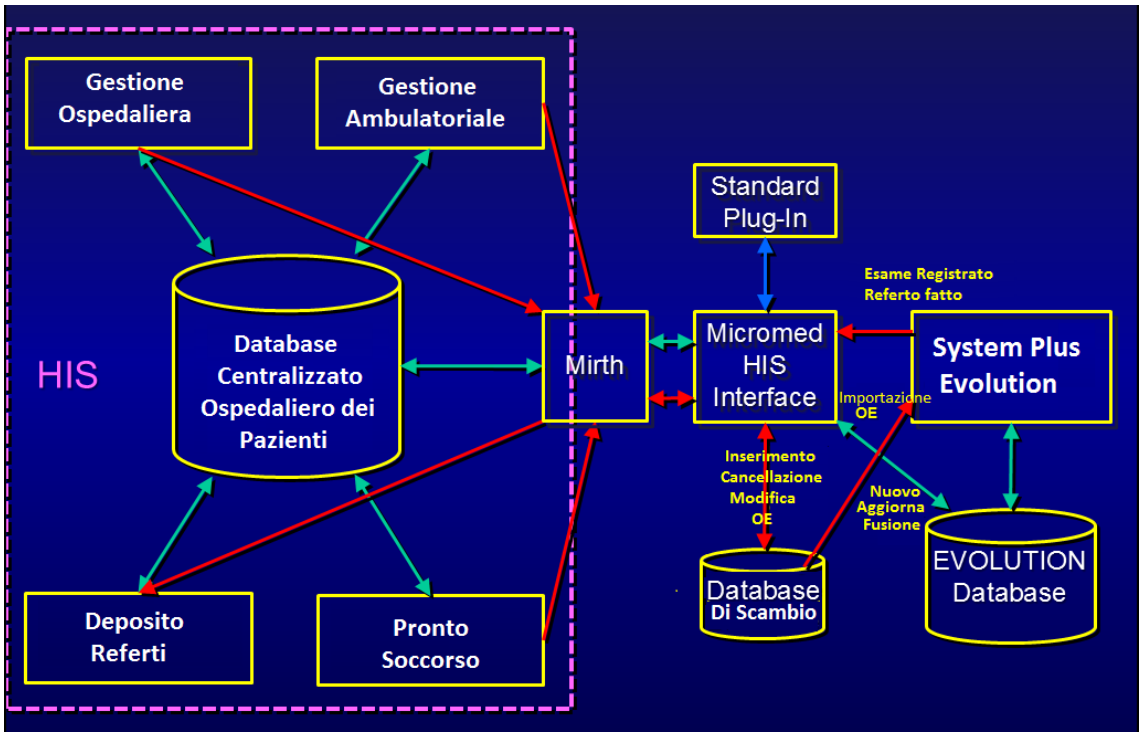

Figura 1.7: Diagramma del flusso informativo fra HIS e System Plus Evolution tramite Mirth Connect

Capitolo 2

Interfacciamento fra SIS e Device Periferici

La struttura tipica di un SIS, come già detto, è quella distribuita. Ciò comporta la necessità di integrare un device periferico, quale il software utilizzato nel reparto di neurofisiologia, nella rete LAN del SIS in modo da assicurare l'interoperabilitá fra il SIS ed il reparto di neurofisologia, tale esigenza é ben descritta da Bracchi Giampio etc. in *Sistemi informativi d'impresa*[2]. Bisogna quindi creare un'interfaccia di comunicazione fra il dispositivo client, reparto di neurofisiologia, ed il server del SIS. I protocolli cui quest'interfaccia deve sottostare, per assicurare l'interoperabilitá, sono quelli discussi nella sezione 1.4, dove per protocollo si intendono quella serie di regole e standard che definiscono i messaggi che verranno scambiati nell'interfaccia. Di seguito vedremo quali sono le componenti del SIS che partecipano all'interfaccia con Mirth e System Plus Evolution, i diagrammi di flusso ed i messaggi tipici che vengono scambiati.

2.1 Gli attori nell'interfaccia

Le piattaforme coinvolte nell'interfaccia, per l'HIS, di cui alcune sono visibili in figure 2.1, sono:

- il Database Ospedaliero dei Pazienti,
- il software per la gestione ambulatoriale,
- il software per l'accettazione,
- il *CUP*,
- altri software dipartimentali, quali il pronto soccorso,
- il deposito per i referti (*Repository*).

Si vedrá poi che le componenti dell'HIS che, in pratica, sono state considerate nello sviluppo dell'interfaccia fra System Plus Evolution e l'HIS sono il CUP ed il Repository poiché sono i dispositivi a cui dobbiamo fare riferimento, rispettivamente, per la ricezione delle richieste d'esame e per l'invio del referto.
Come già detto nel capitolo 1, System Plus Evolution è un software con il quale gli operatori sanitari possono effettuare gli esami, EEG, PE ed EMG ai pazienti e

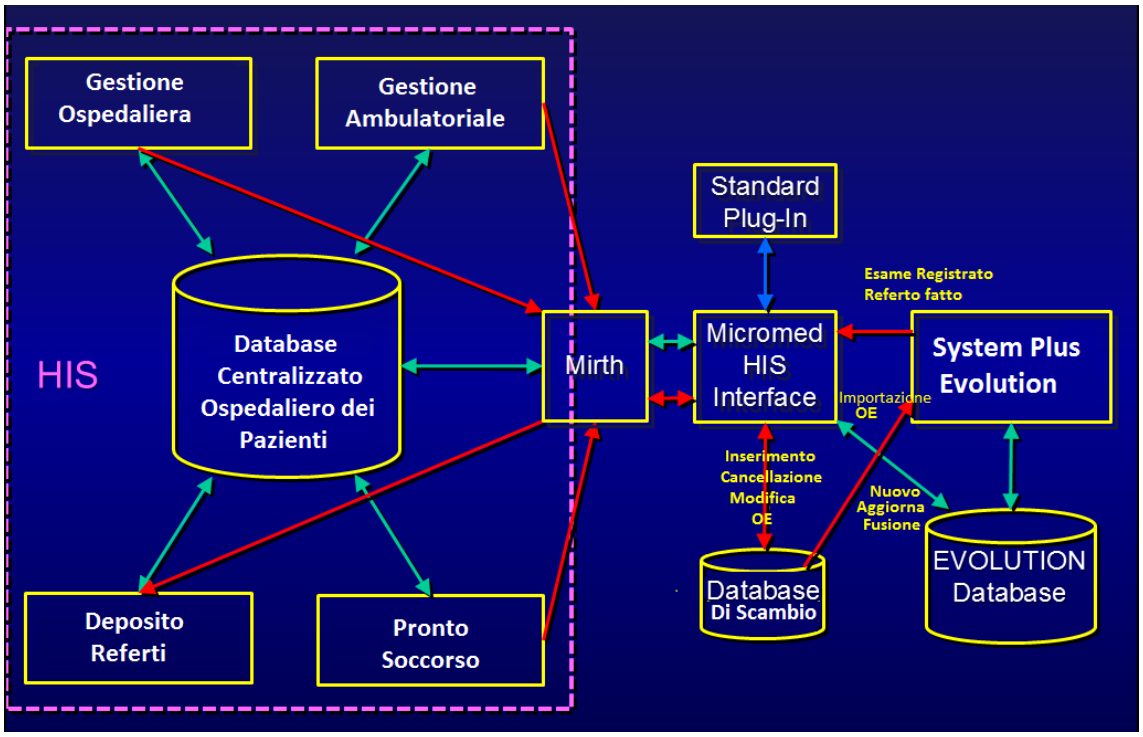

Figura 2.1: Piattaforme coinvolte nell'interfaccia fra HIS e System Plus Evolution tramite Mirth Connect

redigere i referti. Ma il dispositivo preposto alla comunicazione con l'esterno è la Micromed HIS Interface. Quest'ultima svolge essenzialmente due compiti:

1. A fronte della ricezione di un messaggio da parte dell'HIS, la Micromed HIS Interface lo traduce in una serie di istruzioni, utilizzando l'Exchange Database, con le quali, a seconda della tipologia di messaggio, o inserisce la richiesta nel database di System Plus Evolution o aggiorna i dati di un paziente presente nel database. Successivamente invia un messaggio di Acknowledgment all'HIS.

2. A fronte dell'invio di un referto da parte di System Plus Evolution all'HIS, costruisce il messaggio contenente il documento e lo invia al Repository.

Come probabilmente si è già intuito nella realizzazione dell'interfaccia, dal lato Micromed è stata presa in considerazione solo la Micromed HIS Interface, poichè è il dispositivo preposto alla comunicazione con l'esterno.
Poichè l'obiettivo della tesi è la valutazione di Mirth Connect come middleware, è da considerare anche quest'ultimo come attore nell'interfaccia.
Riassumendo, gli attori di cui si è tenuto conto nello sviluppo dell'interfaccia fra HIS e System Plus Evolution sono:

- CUP,
- Repository,
- Mirth Connect,
- Micromed HIS Interface.

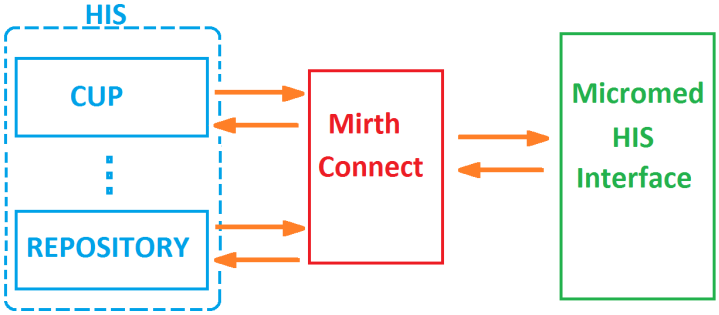

Figura 2.2: Attori nell'interfaccia

2.2 Diagrammi di flusso

Nello sviluppo di un'interfaccia è necessario tenere conto dei protocolli con cui vengono scambiate le informazioni fra i dispositivi. La Micromed HIS Interface comunica con l'esterno attraverso protocolli TCP/IP.
La suite di protocolli TCP/IP fa riferimento a due livelli dello standard ISO-OSI di comunicazione. In particolare il protocollo IP (Internet Protocol) fa riferimento al livello 3, quello di Rete, ed ha il compito di assegnare un indirizzo IP ad ogni terminale che si connette alla rete. Il protocollo TCP (Trasmission Control Protocol) fa riferimento al livello 4, quello di Trasporto, ed ha il compito di rendere affidabile la trasmissione dei dati fra il sitema sorgente e quello in destinazione.
In generale, perchè il dispositivo sorgente possa inviare dati a quello in destinazione sono necessari l'indirizzo IP del terminale in destinazione ed il numero della porta nella quale è in ascolto.
In figura 2.3 sono riportate un esempio delle porte attraverso cui i vari dispositivi comunicano.

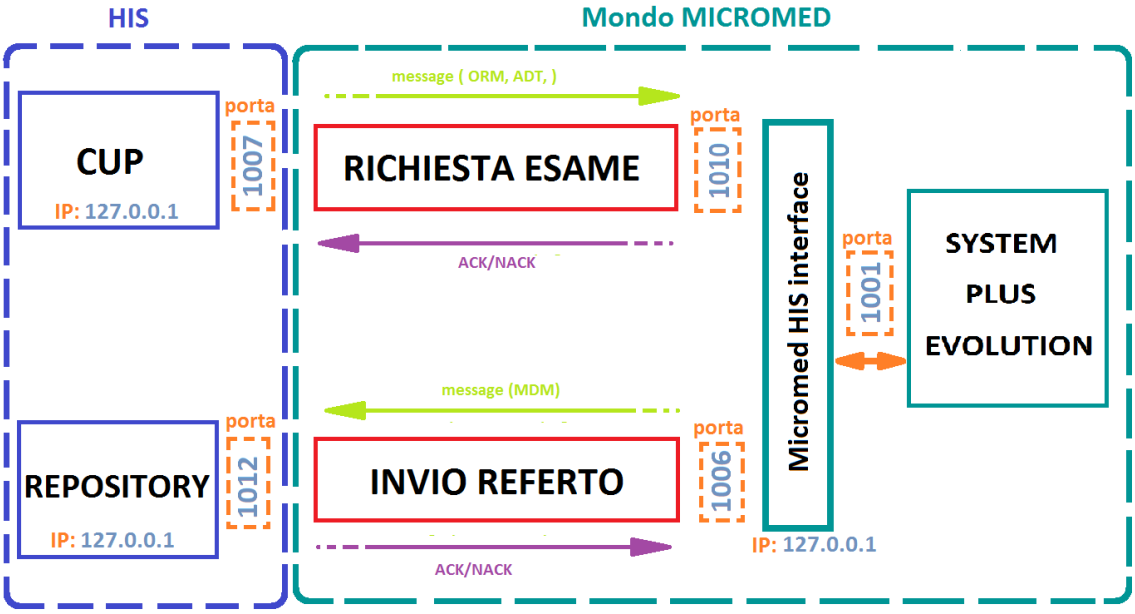

Figura 2.3: Diagramma dell'interfaccia fra System Plus Evolution ed HIS, con dispositivi e le porte tramite cui comunicano

In figura 2.3 è rappresentato il diagramma di flusso fra le componenti dell'HIS, in blu, dei canali del Mirth, in rosso, ed i dispositivi Micromed, in verde. Si può notare immediatamente che sono state inserite due canali Mirth, uno per la consegna delle richieste d'esame ed uno per la consegna dei referti, le cui funzionalità verranno spiegate nel dettaglio nei capitoli successivi. Il motivo, invece, per cui sono state inserite dentro al *mondo Micromed* si sostanzia nel fatto che la gestione dei canali Mirth è completamente demandata all'azienda e non all'ospedale in cui viene installata l'interfaccia.
Dalla figura si può evincere, inoltre, come è distribuito il flusso di messaggi nell'interfaccia. Si distinguono due flussi principali, motivo della creazione di due canali Mirth:

- il primo flusso è dall'HIS verso System Plus Evolution:

  - il CUP invia un messaggio, con protocollo TCP/IP, sulla porta di comunicazione col canale RICHIESTA ESAME (istanza del Mirth). Nell'esempio in figura la porta d'ingresso del canale RICHIESTA ESAME è la 1007.
  - il canale compie, sul messaggio in entrata, le operazioni di filtraggio o trasformazione necessarie e lo invia, sempre utilizzando protocolli TCP/IP, attraverso la porta di comunicazione con la Micromed HIS Interface. Nell'esempio in figura la porta in questione è la 1010.
  - la micromed HIS Interface riceve il messaggio e lo traduce in una serie di istruzioni con le quali scrive nel database di System Plus Evolution. Dopodichè, invia un messaggio di Acknowledgment di avvenuta ricezione del messaggio al canale RICHIESTA ESAME sempre attraverso la porta con cui sono connessi.
  - il canale Mirth trasmette il messaggio di Acknowledgment al CUP tramite la porta con cui sono connessi
  - il CUP riceve l'Acknowledgment e, se necessario, reinvia il messaggio.

- Il secondo flussso è da System Plus Evolution verso l'HIS:
  - viene generato un referto in System Plus Evolution che comunica, attraverso la porta con cui sono connessi, alla Micromed HIS Interface che è pronto per essere esportato
  - la Micromed HIS Interface genera il messaggio, in formato HL7, da inviare all'HIS includendo il referto e lo trasmette sulla porta di comunicazione con il canale INVIO REFERTO, seconda istanza del Mirth. In figura 2.3 la porta con la quale Micromed HIS Interface e canale Mirth comunicano è la 1006.
  - il canale INVIO REFERTO riceve il messaggio e lo invia al REPOSITORY tramite la porta con la quale sono connessi. Nell'esempio in figura tale a porta è la 1012.
  - il REPOSITORY una volta ricevuto il messaggio invia un messaggio di Acknowledgment al canale Mirth
  - il canale Mirth riceve l'acknowledgment, lo trasmette alla Micromed HIS Interface e se necessario mette in coda il messaggio per reinviarlo, successivamente, al REPOSITORY

2.3 Tipologia di messaggi

Nella sezione precedente sono stati spiegati i flussi dei messaggi da e verso la Micromed HIS Interface. Vediamo ora che tipo di messaggi tipicamente vengono scambiati nell'implementazione di un'interfaccia.
Principalmente i messaggi HL7 che vengono scambiati sono quelli di tipo ADT, MDM, ACK ed ORM.

1. i messaggi di tipo **ACK** sono i messaggi di Acknowledgment. Vengono dunque utilizzati per segnalare se il messaggio è stato accettato, rifiutato o filtrato. Il contenuto del messaggio varia, a seconda del tipo di informazione che si vuole trasmettere, in questo modo:

- *Messaggio accettato:* il messaggio contiene un segmento MSH ed un segmento MSA. Si riconosce l'avvenuta accettazione del messaggio dal codice AA (Application Accepted) nel segmento MSA. Si può vedere un esempio del messaggio in figura 2.4.

```
MSH|^~\&|SYSTEMPLUS|MICROMED|OF_EXPRIVIA|EXPRIVIA|20150908103141||ACK|||2.5
MSA|AA|000000004
```

Figura 2.4: Messaggio HL7 di Acknowledgment di tipo AA

- *Messaggio filtrato:* anche in questo caso il messaggio contiene, solitamente, solo 2 segmenti l'MSH e l'MSA. Tale tipo di messaggio è identificabile dal codice AR (Application Rejected) nel segmento MSA. In figura 2.5 se ne può vedere un'esempio.

```
MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20150904170735.169||ACK^T10
^ACK|20150904170735.169|P|2.5
MSA|AR|201508051522|Message Rejected.
```

Figura 2.5: Messaggio HL7 di Acknowledgment di tipo AR

- *Messaggio rifiutato:* In questo caso il messaggio contiene i due segmenti MSH ed MSA, e ció che identifica il fatto che il messaggio è andato in errore è il codice AE (Application Errored) nel segmento MSA. Inoltre, potrebbe essere presente il segmento opzionale ERR nel quale viene riportata la descrizione del motivo per cui il messaggio è andato in errore. In figura 2.6 viene riportato un esempio del messaggio, senza però il segmento ERR, in quanto opzionale

```
MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20150904170735.169||ACK^T10
^ACK|20150904170735.169|P|2.5
MSA|AE|201508051522|Message Errored.
```

Figura 2.6: Messaggio HL7 di Acknowledgment di tipo AE

2. i messaggi di tipo **ORM**- Order Message sono utilizzati come messaggi generici per trasmettere informazioni riguardo una richiesta esame. Possono quindi essere utilizzati per l'invio di nuove richieste e per la modifica o la cancellazione di quelle già esistenti. Tali messaggi devono contenere obbligatoriamente i segmenti MSH, PID, PV1, OBR, ORC. Il segmento MSH è necessario in quanto contiene le informazioni necessarie all'identificazione del messaggio, quali il tipo, il mittente, il destinatario, la data e l'ora della creazione etc.. Il PID contiene, invece, tutte le informazioni anagrafiche del paziente per cui é stato generato il messaggio. Il PV1 contiene le informazioni ambulatoriali relative al paziente. L'ORC identifica, invece, le informazioni relative al tipo di richiesta (nuova, modifica o cancellazione) e del medico che l'ha generata. Infine L'OBR riporta le informazioni proprie dell'esame che bisogna effettuare, quali il tipo d'esame e l'ora. In figura 2.7 viene riportato un esempio di una richiesta d'esame.

```
MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20140114151219||ORM^O01|00
0000004|P|2.5|||||8859/1|

PID|1||259800^^^PK||PROVAprova^BOLOGNA^^^^^L||19710101|F|||VIA RESIDENZA
10^^MILANO^MI^20100^^L^^015146~VIA DOMICILIO
16^MI^MILANO^^20103^^H^^015146~^^MILANO^^^^N^^015146||02787881^^^PIPPO@PLUT
O.COM~3409181881|||||PRVGNS71A41F205Q|CS000112123|||||||100^ITALIANA|

PV1|1|O|||||||||||||||||||||||||||||||||||||||||||||||||||||A

ORC|NW|000000003||0000001|||1^^^^^R||20140114151219|||MZZGPP75A01G224F^MAZZI
NI^GIUSEPPE^^^^^^CF^L^^^08^|

OBR|1|000000101||0001^EEG|||||||||QUESITO
CLINICO|||||0502568999|||||||||||WALK||||||20140317120000|
```

Figura 2.7: Messaggio HL7 per una richiesta esame

3. i messaggi di tipo **ADT** possono essere generati in funzione di diversi eventi scatenanti, quali l'inserimento di un nuovo paziente nel database, l'aggiornamento dei dati o la fusione di due pazienti. Nel campo *Message type* nel segmento MSH viene riportato il codice della tipologia del messaggio che il segmento identifica, ad esempio ADT, seguita dal codice dell'evento per cui é stato generato, ad esempio A31. Il codice relativo all'evento è separato dal codice relativo al tipo di messaggio dal carattere utilizzato come separatore di componenti, ad esempio^. In base all'evento scatenante il messaggio deve contenere obbligatoriamente alcuni segmenti. Di seguito viene presentata una lista degli eventi, nell'ambito ADT, piú frequenti e le loro caratteristiche:

    - *Inserimento di un nuovo paziente:* questa tipologia di eventi è identificata dal codice A01 e deve contenere obbligatoriamente i segmenti MSH, EVN, PID e PV1. Il segmento EVN contiene informazioni sull'evento che ha generato il messaggio, quali il codice dell'evento, l'ora in cui si è verificato, etc. In figura 2.8 se ne può vedere un esempio.

```
MSH|^~\&|SUNS1|OPN01|AZIS|CMD|200607261151||ADT^A01|1342749200|P|2.2

EVN|A01|200607270800

PID||4509276047^^58276|0187214^51286||JOHNSON^MARINA||19451027|F|||To
wnstreet 53PARIS8800^150||777/789456~0476
357737||N|W|||45092711880||""^^""|||||B

PV1||O|5501^0113^02|U|00060292||00276^DELBARE^POL^^DR.|00276^DELBARE^P
OL^^DR.||1901|||N|01||||""|0161782703
```

Figura 2.8: Messaggio HL7 di tipo ADT relativo all'inserimento di un paziente

    - *Aggiornamento dei dati di un paziente:* tale tipologia viene identificata dai codici A31 o A08. I messaggi relativi a questo evento devono contenere obbligatoriamente i segmenti MSH, EVN, PID, PV1. Un esempio è riportato in figura 2.9. Sono solitamente utilizzati per informare il sistema sugli spostamenti del paziente interni all'ospedale, per esempio cambio reparto, letto, ma anche per modificare dati esterni, quali il cambio di residenza, numero di telefono etc.,

```
MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20140114151219||ADT^A31|00
0000003|P|2.5|||||8859/1|

EVN|A31|20140114151219|

PID|1||259856^^^PK||PROVA^AGNESE^^^^^L||19710101|F|||VIA RESIDENZA
10^^MILANO^MI^20100^^L^^015146~VIA DOMICILIO
16^MI^MILANO^^20103^^H^^015146~^^MILANO^^^^N^^015146||02787881^^^PIPPO@PLUT
O.COM~3409181881|||||PRVGNS71A41F205Q|CS000112123|||||||100^ITALIANA|

PV1||N|
```

Figura 2.9: Messaggio HL7 di tipo ADT relativo all'aggiornamento dei dati di un paziente

- *Fusione di due pazienti:* può succedere che, a causa di disattenzioni nell'inserimento dei dati di un paziente, siano presenti due pazienti che in realtà sono la stessa persona. Per operare la fusione di questi due pazienti si usano messaggi, il cui codice evento è l'A40, che devono contenere obbligatoriamente i segmenti MSH, EVN, PID e MRG. In figura 2.10 ne viene riportato un esempio. In particolare é necessaria la presenza del segmento MRG poichè è il segmento che contiene le informazioni relative ai due pazienti da fondere.

```
MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20140114151219||ADT^A40|00
0000002|P|2.5|||||8859/1|

EVN|A40|20140114151219|

PID|1||259856^^^PK||PROVA^AGNESE^^^^^L||19710101|F|||VIA RESIDENZA
10^^MILANO^MI^20100^^L^^015146~VIA DOMICILIO
16^MI^MILANO^^20103^^H^^015146~^^MILANO^^^^N^^015146||02787881^^^PIPPO@PLUT
O.COM~3409181881|||||PRVGNS71A41F205Q|CS000112123|||||||100^ITALIANA|

PV1||N|

MRG|36525^^^PK|
```

Figura 2.10: Messaggio HL7 di tipo ADT utilizzato per la fusione di due pazienti

4. i messaggi di tipo **MDM** vengono generati per fungere da vettori di documenti o informazioni importanti relative ai pazienti. Anche questa tipologia di messaggi può essere generata da diversi eventi. In particolare gli eventi che possono dare luogo ad un messaggio di tipo MDM sono 11, quindi dal T01 al T11. Questi undici eventi sono ulteriormente divisi in due categorie, gli eventi col numero pari sono adibiti al trasporto di documenti, mentre quelli dispari identificano solo il traporto di informazioni contenute nel messaggio. Poichè System Plus Evolution, una volta effettuato un esame, genera un referto in formato pdf di seguito vediamo i due eventi, con codice pari, per cui la Micromed HIS Interface genera i messaggi:

   - *Invio di un nuovo referto:* in questo caso l'evento scatenante ha codice T02. I messaggi relativi a questo evento devono contenere i segmenti MSH, EVN, PID, PV1, TXA, OBX. Il segmento TXA contiene informazioni relative al documento che è stato redatto ma non ne contiene le

informazioni, compito invece svolto dal segmento OBX che incorpora il referto codificato come si può vedere in figura 2.11

MSH|^~\&|SYSTEMPLUS|MICROMED|OF_EXPRIVIA|EXPRIVIA|20150803115447||MDM^T02|201508031154|P|2.5||||||8859/1

EVN|T02

PID|1||1636690^^^PK^^^PK||BARBIERI^GIULIA^^^^^L||20141122|F|pippo||VIA CALAMOSCO,10^^BOLOGNA^BO^40100|||||||BRBGLI14S62A944S

PV1|1|O|||||||||||||||||||||||||||||||||||||||||||||||||||||A

ORC|RE|8585147||0000001|CM||||20150803110300|||^MAIDA^WALTER PASQUALE

OBR|1|||89191^ELETTROENCEFALOGRAMMA CON VIDEOREGISTRAZIONE|||||||||||||||||||||F||||||||||20150803110000

TXA|1|HP|TEXT|||20150803114300|||MRCVNT61B53A944G^^^^^^^^CF|||8585147_0|||||AU||AV

OBX|1|ED|00030000450003000014||^application^TEXT^Base64^VGVjbmljbyBlc2VjdXRvcmU6IFViZXJ0aWVsbG8gCiBFRUdzdGFuZGFyZCArIEVDRyArIEVNRzEgKGRlbHRkeCkKIAogMDMvMDgvMjAxNSBFc3Rlcm5vCiBQcmltbyBFRUucgdmVnbGlhIGluIHB6IGNoZSBsYSBzY29yc2Egc2V0dGltYW5hIGhhIHByZXNlbnRhdG8gZXBpc29kaSBkaSBpcnJpZ2lkaeW1lbnRvIGRlZ2xpIGFydGkgaW4gbW9tZW50aSBkaSBlY2NpdGFtZWs50by4gTmVzc3VuYSB0ZXJhcGlhIGluIGNvcnNvLgogCiBBdHRpdml0w6AgYmlvZWxldHRyaWNhIHNpbW1ldHJpY2EgLCBsaWV2ZW1lbnRlIGUgZGlmZnVzYW1lbnRlIHJhbGxlbn

Figura 2.11: Messaggio HL7 di tipo MDM per l'invio di un nuovo referto

- *Versionamento di un referto esistente:* in tal caso l'evento ha codice T10. Anche questa tipologia deve contenere obbligatoriamente i segmenti MSH, EVN, PID, PV1, TXA, OBX. Nel segmento OBX é contenuto il referto, codificato, che deve andare a sostituire quello esistente. In figura 2.12 se ne può vedere un esempio.

MSH|^~\&|SYSTEMPLUS|MICROMED|OF_EXPRIVIA|EXPRIVIA|20150805152222||MDM^T10|201508051522|P|2.5||||||8859/1

EVN|T10

PID|1||1518286^^^PK^^^PK||PONTELLINI^ENRICO^^^^^L||19980226|M|||VIA TUMIATI, 66/1^^PESARO^PU^61100|||||||PNTNRC98B26A944D

PV1|1|O|||||||||||||||||||||||||||||||||||||||||||||||||||||A

ORC|RE|8514039||0000001|CM||||20150617123100|||^MARCHIANI^VALENTINA

OBR|1|||89191^ELETTROENCEFALOGRAMMA CON VIDEOREGISTRAZIONE|||||||||||||||||||||F||||||||||20150617115000

TXA|1|HP|TEXT|||20150617125500|||MSCFMN55S68A463Y^^^^^^^^CF|||8514039_1|8514039_0||||AU||AV

OBX|1|ED|00030000450003000015||^application^TEXT^Base64^MTdcNlwxNSAgRXN0ZXJubwogCiBQZWRyYXp6aQogCiBFRUfcgdmVnbGlhLlVsdGltZSBhc3NlbnplIDIwMTMuVGVyYXBpYTpEZXBha2luIDEyMDAgbWscuCiAKIAogVHJhY2NpYXRvIHJlZ2lzdHJhdG8gY29uIGJ1b25hIGNvbGxhYm9yYXppb25lIGRlbCBnaW92YW5lLgogCiAKIFRhbGUgdHJhY2NpYX

Figura 2.12: Messaggio HL7 di tipo MDM per il versionamento di un referto esistente

Capitolo 3

Il middleware Mirth Connect

3.1 Introduzione

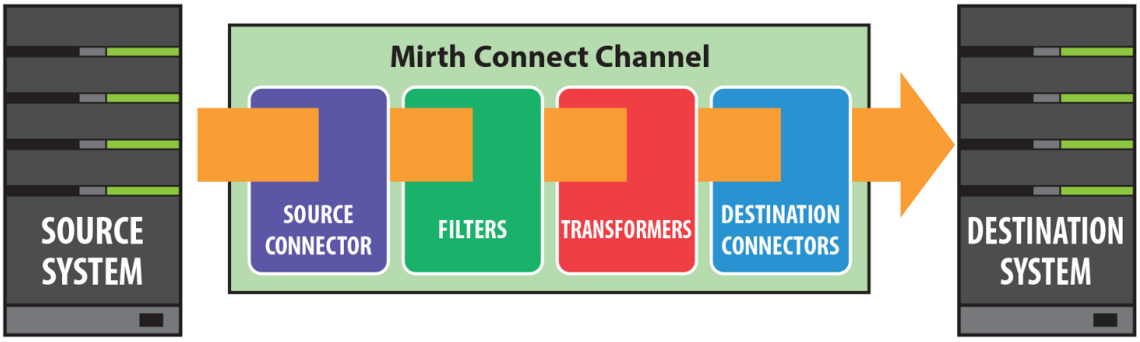

Figura 3.1: Flusso generico del messaggio all'interno del canale Mirth

Mirth connect è un sistema d'integrazione sanitaria *open source* il cui scopo è la creazione di interfacce, o canali, che permettono il filtraggio, la trasformazione ed il routing dei messaggi attraverso diversi sistemi informativi in modo che possano condividere informazioni fra loro, per approfondimenti vedi `www.mirth.com`. La ricezione e l'invio delle informazioni può avvenire attraverso diversi tipi di protocolli: TCP/IP, HTTP, Web Services (SOAP), Database, File System, Email.

Inoltre Mirth Connect supporta diversi formati per le informazioni: HL7 v2.x, HL7 v3, XML, DICOM, PDF, RTF.
Come un'interprete che traduce una lingua straniera, Mirth Connect traduce i messaggi inviati da un sistema sorgente in maniera che siano comprensibili al sistema destinazione. Tale funzionalitá si può raggiungere tramite le seguenti fasi, che sono disponibili in ogni canale:

- **filtraggio:** Mirth Connect legge i parametri del messaggio ed in base a questi può far passare il messaggio alla fase di trasformazione o filtrarlo.
- **trasformazione:** Mirth Connect converte il messaggio arrivato in un messaggio che abbia gli standard richiesti dal sistema destinazione per poter essere eseguito correttamente
- **routing:** In questa fase Mirth Connect si assicura che il messaggio arrivi correttamente a destinazione.

Nelle sezioni successive cercheremo di spiegare come poter creare questi canali a seconda delle esigenze richieste.

3.2 Fondamenti del Mirth

3.2.1 Workflow del messaggio nel canale

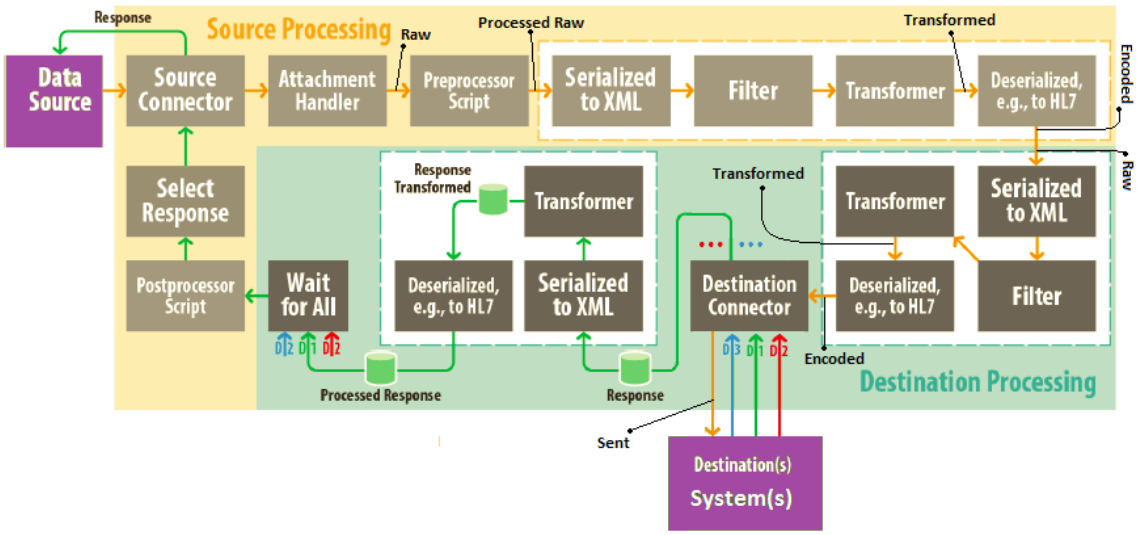

Figura 3.2: Workflow del messaggio all'interno del canale Mirth

Come si può vedere in figura 3.1 utilizzare Mirth Connect si traduce nell'inserire un canale fra il sistema Sorgente e quello Destinazione. Il Canale Mirth è suddiviso in due connettori: quello Sorgente, il cui compito è la corretta ricezione del messaggio, e quello Destinazione, il cui compito è portare a termine la consegna del messaggio. Ogni canale deve contenere un solo connettore Sorgente, ma può avere più connettori Destinazione, ognuno dei quali invia il messaggio ad un sistema diverso, che compiono ciascuno delle operazioni di filtraggio e/o trasformazione differenti da connettore a connettore in base alle esigenze del sistema a cui il connettore deve inviare il messaggio.
Il processing di un messaggio nel canale coinvolge molte attività e viene dunque suddiviso in una serie di passi divisi fra i connettori Sorgente e Destinazione. In figura 3.2 è possibile vedere uno schema a blocchi nel quale sono suddivise le varie fasi che passa il messaggio, sia nel connettore Sorgente, riquadri contenuti nella parte in giallo, sia nel connettore Destinazione, riquadri contenuti nella parte in verde. Per il connettore Sorgente il workflow del messaggio è il seguente:

1. Il messaggio viene ricevuto dal connettore sorgente
2. (Opzionale - dev'esserci un'allegato) Se nel messaggio c'è un allegato, questo viene estratto e/o salvato
3. Dopo il prelievo dell'allegato il messaggio viene considerato **Raw**
4. (Opzionale) Viene eseguito il *Preprocessor* script sul messaggio Raw e viene salvato come **Processed Raw**
5. Il messaggio Raw (od il Processed Raw) viene serializzato (convertito) in formato XML
6. (Opzionale - dev'esserci un filtro) Viene eseguito il filtro sul messaggio in formato XML

7. (Opzionale - dev'esserci un trasformatore) Viene eseguito il trasformatore sul messaggio in formato XML

8. (Opzionale - dev'esserci o un filtro o un trasformatore) Il messaggio, dopo che è stato eseguito il filtro/trasformatore, viene salvato come **Transformed**

9. (Opzionale - dev'esserci o un filtro od un trasformatore) Il messaggio XML viene convertito nel tipo di messaggi in uscita (HL7,EDI,etc) e salvato come **Encoded**. Dopodichè il messaggio encoded viene passato al connettore Destinazione.

10. Dopo che tutti i connettori destinazione sono stati eseguiti, viene eseguito il *Postprocessor* script

11. Viene selezionata una risposta o autogenerata o da uno dei connettori destinazione o dal postprocessor e viene salvata come **Response** del connettore sorgente e viene inviata al sistema sorgente.

Per il connettore destinazione il workflow del messaggio è il seguente:

1. Il messaggio Encoded generato dal connettore sorgente arriva al connettore destinazione e viene salvato come il **Raw** per il connettore destinazione

2. Il messaggio Raw viene serializzato (convertito) in formato XML

3. (Opzionale - dev'esserci un filtro) Viene eseguito il filtro sul messaggio in formato XML

4. (Opzionale - dev'esserci un trasformatore) Viene eseguito il trasformatore sul messaggio in formato XML

5. (Opzionale - dev'esserci o un filtro od un trasformatore) Il messaggio, dopo che è stato eseguito il filtro/trasformatore, viene salvato come **Transformed**

6. (Opzionale - dev'esserci o un filtro od un trasformatore) Il messaggio XML viene convertito nel tipo di messaggi in uscita (HL/,EDI,etc) e salvato come **encoded**

7. Il connettore destinazione costruisce il messaggio da inviare al sistema in destinazione con uno dei messaggi salvati (Raw, Transformed, Encoded) a nostra scelta, lo invia al sistema in destinazione e ne salva il contenuto come **Sent**. La scelta del messaggio da inviare si specifica nei parametri del connettore destinazione.

8. Viene ricevuta la risposta dal sistema in destinazione e salvata come **Response**

9. La risposta viene convertita in formato XML

10. (Opzionale- dev'esserci un trasformatore) Viene eseguito il trasformatore sulla risposta in formato XML

11. (Opzionale- dev'esserci un trasformatore) La risposta trasformata viene salvata come **Response Transformed**

12. (Opzionale- dev'esserci un trasformatore) La risposta in XML viene convertita nel tipo di messaggi in uscita e salvata come **Processed Response**

Come è stato visto, ogni connettore ha diverse versioni del messaggio relative allo step che è stato compiuto:

- ***Sorgente***

  1. **Raw:** il messaggio così com'è arrivato al connettore
  2. **Processed Raw:** il messaggio dopo l'esecuzione del Preprocessor script, se presente
  3. **Transformed:** rappresentazione XML del messaggio dopo l'esecuzione del filtro e/o del trasformatore (presente solo se esiste un filtro od un trasformatore)
  4. **Encoded:** il messaggio così come viene inviato ai connettori destinazione, per esempio in formato HL7
  5. **Response:** il messaggio inviato al sistema sorgente

- ***Destinazione***

  1. **Raw:** messaggio ricevuto dal connettore sorgente, quindi uguale all'encoded del sorgente
  2. **Transformed:** rappresentazione XML del messaggio dopo l'esecuzione del filtro e/o del trasformatore (presente solo se esiste un filtro od un trasformatore)
  3. **Encoded:** il messaggio dopo il trasformatore ma convertito nel tipo in uscita
  4. **Sent:** il messaggio che effettivamente viene inviato al sistema in destinazione
  5. **Response:** il messaggio ricevuto dal sistema in destinazione come risposta all'avvenuta ricezione del messaggio inviato
  6. **Response Transformed:** rappresentazione XML della risposta (presente solo se esiste uno script nell'*Edit Response* nel connettore)
  7. **Processed Response:** la risposta in uscita dal trasformatore convertita nel tipo di messaggio in uscita

3.2.2 Interfaccia Utente

L'interfaccia utente di Mirth Connect Administrator si suddivide nelle seguenti pagine principali:

1. ***Dashboard***
2. ***Channels***
3. ***Users***
4. ***Settings***
5. ***Alerts***
6. ***Events***
7. ***Extensions***

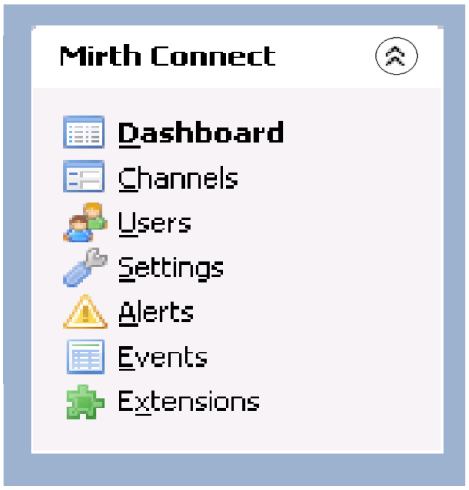

Figura 3.3: Pannello di Mirth Connect

Si accede a ciascuna di queste pagine dal pannello **Mirth Connect** presente nella schermata iniziale del **Mirth Connect Administrator**. In seguito verranno approfondite solamente le pagine 1. e 2. perchè pagine di maggior interesse.

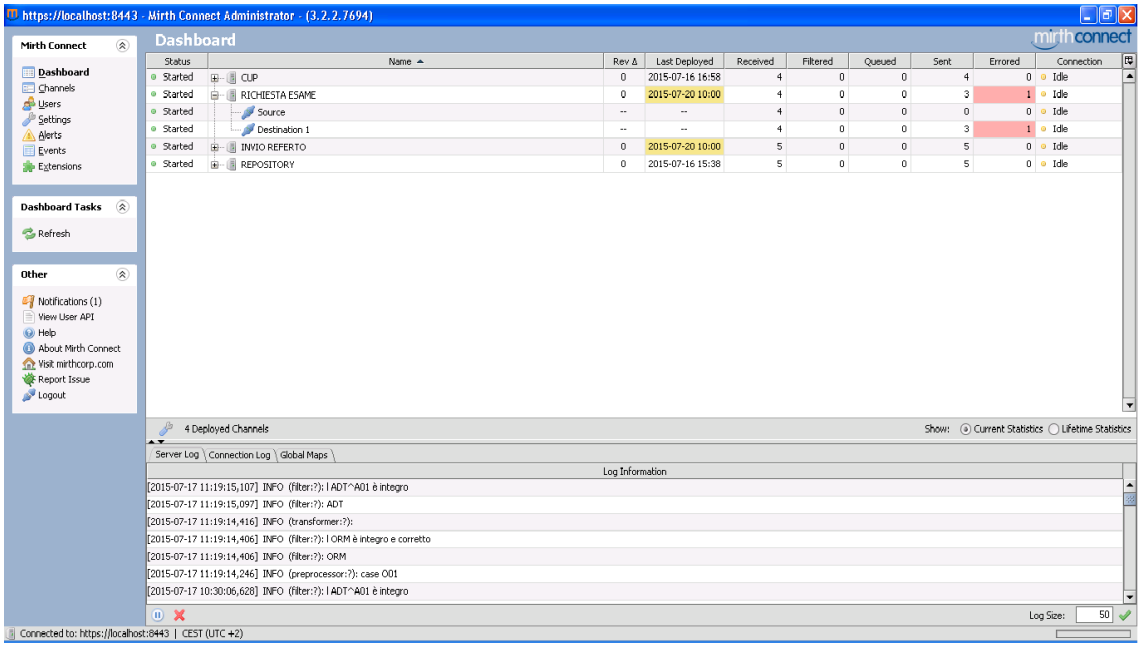

Figura 3.4: Schermata inziale del Mirth: la Dashboard

1. La ***Dashboard*** è la prima schermata che appare all'apertura di Mirth Connect Administrator.

   Da questa pagina è possibile visualizzare i canali attivi e monitorare il loro funzionamento. In particolare ciò che possiamo fare/vedere in questa pagina è:

   - Prendere visione dei canali di cui è stato fatto il Deploy
   - Monitorare lo stato dei canali (*Started/Stopped/Paused/Halted*)
   - Fare l'Undeploy dei canali
   - Prendere visione del Server Log

- Vedere le statistiche dei messaggi sia per ogni canale, sia per ogni singolo connettore
- Accedere alla pagina di *Channel messages* che mostra i messaggi ricevuti ed inviati dal canale selezionato.

Nella schermata ***Dashboard*** compare alla sinistra il pannello **Dashboard tasks** i cui pulsanti hanno il seguente scopo:

- **Refresh:** aggiorna i valori nella lista dei canali
- **Send Message:** permette l'invio di un messaggio nel canale selezionato
- **View Messages:** permette l'accesso alla pagina di *Channel messages* che mostra i messaggi ricevuti ed inviati dal canale selezionato
- **Remove All Messages:** Rimuove i messaggi del canale selezionato

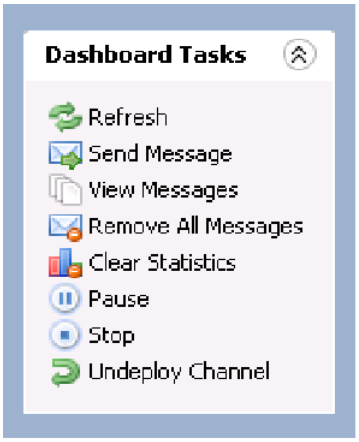

Figura 3.5: Pannello che riporta le funzioni utilizzabili nella Dashboard

- **Clear Statistics:** apre una finestra di dialogo nella quale si può scegliere quali statistiche azzerare nel canale selezionato. Azzerando le statistiche non verranno cancellati i messaggi dallo storico.
- **Pause:** mette in pausa il canale selezionato (è presente solo se lo stato del canale selezionato è: *Started*)
- **Start:** avvia il canale selezionato (è presente solo se lo stato del canale selezionato è: *Stopped/Paused*)
- **Stop:** ferma il canale selezionato dopo aver finito di processare il messaggio corrente (è presente solo se lo stato del canale selezionato è: *Started/Paused*)
- **Halt:** ferma immediatamente il canale selezionato lasciando incompleti i messaggi iniziati. Una volta riavviato il canale, il messaggio riprende dall'ultimo stadio che era stato completato prima del Halt (è presente solo se lo stato del canale selezionato è: *started/Paused/Stopped*).
- **Undeploy Channel:** rimuove il canale selezionato dalla dashboard. Il canale non riapparirà nella dashboard finche non verrà fatto il deploy dalla pagina ***Channels***.

2. La ***Channels*** è la pagina in cui si può prendere visione di tutti i canali creati/importati, anche quelli di cui non è stato fatto il Deploy.

   In particolare ciò che possiamo fare/vedere in questa pagina è:

   - Importare/Esportare i canali
   - Prendere visione dello stato dei canali(*Enabled/Disabled*)
   - Fare il Deploy dei canali

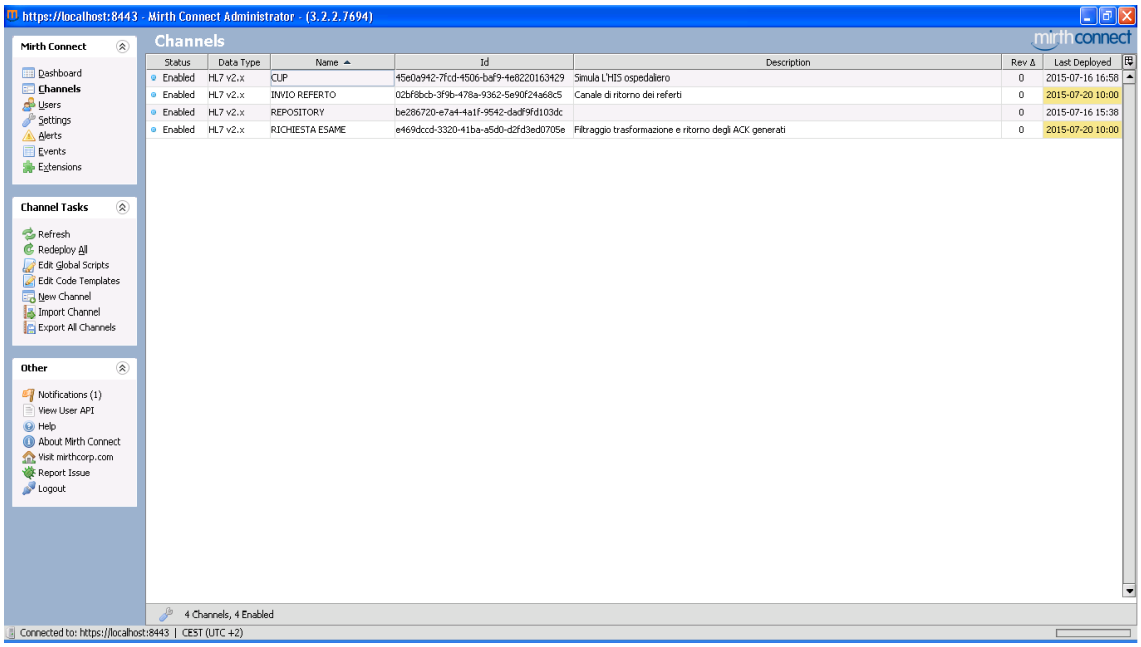

Figura 3.6: Schermata Channels

- Accedere alla pagina *Edit Channels* per creare un nuovo canale o modificarne uno già esistente

In questa pagina compare alla sinistra il pannello **Channel tasks** i cui pulsanti hanno il seguente scopo:

- **Refresh:** aggiorna i valori nella lista dei canali
- **Redeploy All:** fa il Redeploy di tutti i canali
- **Deploy Channel:** fa il Deploy del canale selezionato
- **Edit Global Script:** permette l'editing di uno script che non è specifico di un canale ma vale per tutti i canali presenti, e viene eseguito subito prima degli script specifici dei canali. Lo script può essere scritto per le fasi di: Deploy, Undeploy, Preprocessor, Postprocessor.
- **Edit Code Templates:** permette la creazione e l'inserimento di function da poter poi utilizzare nei canali
- **New Channel:** permette l'accesso alla pagina *Edit Channel* per la creazione di un nuovo canale
- **Import Channel:** apre una finestra di dialogo in cui bisogna selezionare il canale da importare
- **Export All Channels:** apre una finestra di dialogo dove bisogna selezionare la cartella in cui verranno esportati tutti i canali presenti nella pagina ***Channels***
- **Export Channel:** apre una finestra di dialogo dove bisogna selezionare la cartella in cui verrà esportato il canale selezionato

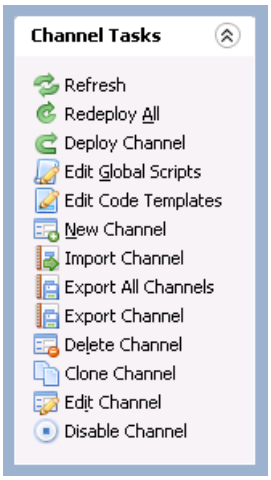

Figura 3.7: Pannello delle funzioni utilizzabili nella Channels

- **Delete Channel:** elimina definitivamente il canale selezionato. Se il canale è già avviato, verrà prima fatto l'Undeploy.
- **Clone Channel:** clona il canale selezionato
- **Edit Channel:** permette l'accesso alla pagina *Edit Channel* per la modifica del canale selezionato
- **Disable Channel:** disabilita il canale selezionato. Non é possibile fare il Deploy se il canale è disabilitato (è presente solo se il canale selezionato è: *Enabled*)
- **Enable Channel:** abilita il canale selezionato (è presente solo se lo stato del canale selezionato è: *Disabled*)

3.3 L'editing di un canale

L'editing di un canale avviene attraverso l'*Edit Channel*, accessibile dal pannello nella schermata ***Channels***, e si compone di 4 schermate:

1. *Summary*
2. *Source*
3. *Destinations*
4. *Scripts*

In ogni connettore sorgente e destinazione è possibile fare sul messaggio delle operazioni di filtraggio e di trasformazione, che vengono definite, rispettivamente, con l'*Edit Filter* e l'*Edit Transformer*. Inoltre è possibile, solo nel connettore destinazione, compiere delle operazioni di trasformazione sulla risposta, messaggi di Acknowledgment, che si riceve dal sistema al quale si è inviato il messaggio. Tale funzionalità viene definita nell'*Edit Response*.
Di seguito verranno presentate (non nel dettaglio) tali strumenti che sono utili ed essenziali per la creazione di un canale.

3.3.1 Summary

La scheda Summary ci permette di definire i parametri generici del canale quali il nome, lo stato iniziale, la quantità di dati da salvare nel Database e ogni quanto pulirlo, i metadati da generare per ogni messaggio e una descrizione del canale. La scheda si divide in 6 riquadri ognuno dei quali ha una certa specificità:

1. **Channel Properties**

   In questo riquadro è possibile definire le proprietà generali del canale.

   - *Name:* Si inserisce il nome da dare al canale
   - *Initial State:* bisogna selezionare lo stato iniziale (STARTED, STOPPED, PAUSED) che deve avere il canale dopo ogni Deploy.
   - *Enabled:* Se viene spuntata questa opzione, il canale viene abilitato ed é quindi possibile farne il deploy; se disabled non è possibile.

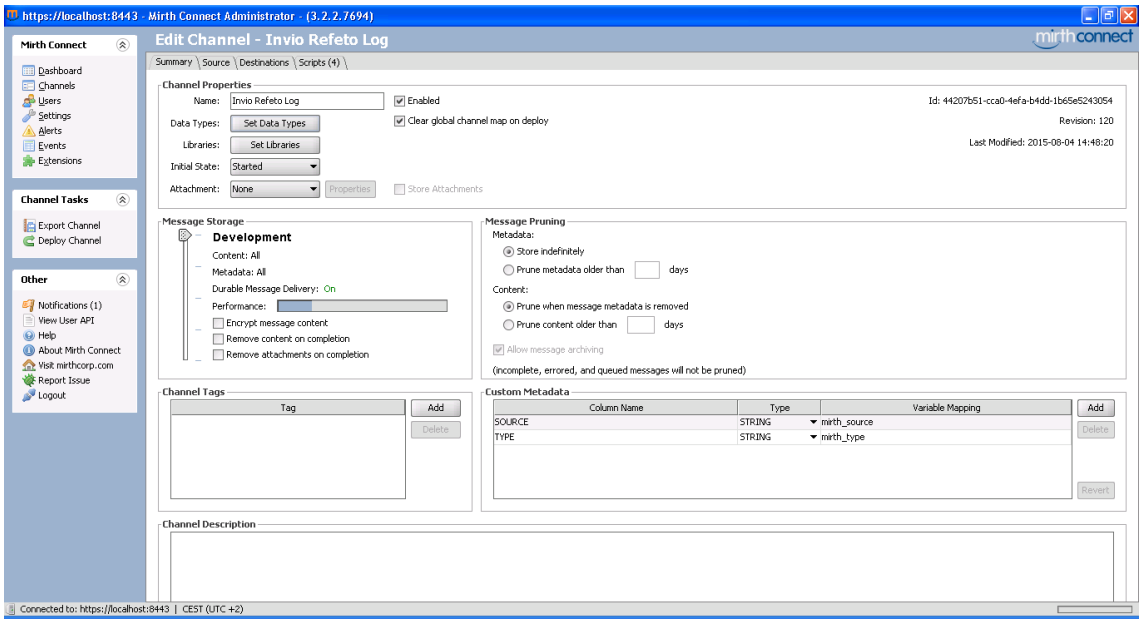

Figura 3.8: Scheda Summary dell'Edit Channel

- *Clear global channel map on deploy:* permette di pulire la global channel map ad ogni deploy. Tale opzione è abilitata di default.
- *Data Type:* si possono qui selezionare le proprietà dei messaggi in arrivo ed in uscita a livello di singolo connettore. In particolare si può definire il tipo e le proprietà dei messaggi in entrata:

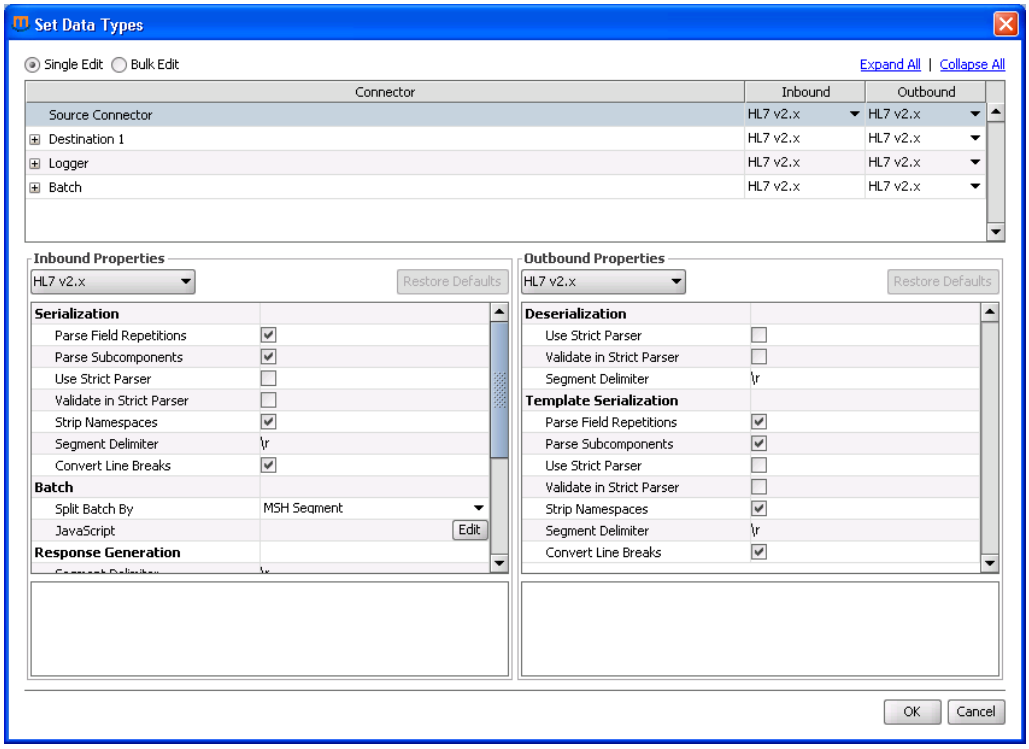

Figura 3.9: Finestra del Set Data Types

  - La *Serializzazione* del messaggio. Qui si può definire la maniera in cui serializzare i messaggi in entrata al canale

- *Batch*. Se il canale gestisce file batch si puó definire in base a che parametro i messaggi nel file vengono individuati
- Nella *Response Generation* si possono definire i codici ed i messaggi con i quali la risposta autogenerata del Mirth viene composta

Si possono definire anche il tipo e le proprietà dei messaggi in uscita:

- La *Deserializzazione* del messaggio. Serve a definire la maniera in cui il messaggio deve essere deserializzato
- Nella *Template Serialization* si definiscono i parametri di serializzazione dell'Outbound Template

Si possono inoltre vedere l'Id del canale, il numero di modifiche effettuate e la data dell'ultima modifica .

2. **Message Storage**

   In questo riquadro é possibile selezionare la quantità di messaggi che devono essere salvati nel database. Bisogna allora decidere fra 5 livelli, piú in basso andiamo piú alte saranno le performance del canale poichè saranno minori i dati che verranno salvati. I livelli sono:

   - Development: salva tutti i messaggi prodotti durante il processing (Encoded, Transformed, Raw, Response, etc...) e tutte le mappe ed i Metadata, le performance di questo canale sono le più basse
   - Production: salva solo i messaggi di tipo Encoded, Sent, Raw, Response tutte le mappe e tutti i Metadata
   - Raw: salva solo i messagg di tipo Raw e tutti i metadata
   - Metadata: salva solo i metadati generati da ogni messaggio
   - Disabled: non salva alcun tipo di dato e per questo è la modalità con le performance migliori.

3. **Message Pruning**

   In questo riquadro si può selezionare ogni quanto pulire la memoria dai messaggi e dai metadati salvati.

4. **Tag**

   Permette di inserire dei tag al canale

5. **Custom Metadata**

   In questo riquadro è possibile selezionare una particolare variabile a noi utile che viene estratta da ogni messaggio che passa nel canale per poterla visualizzare in maniera immediata nel *view message*.

6. **Channel Description**

   Permette di dare una descrizione sullo scopo del canale

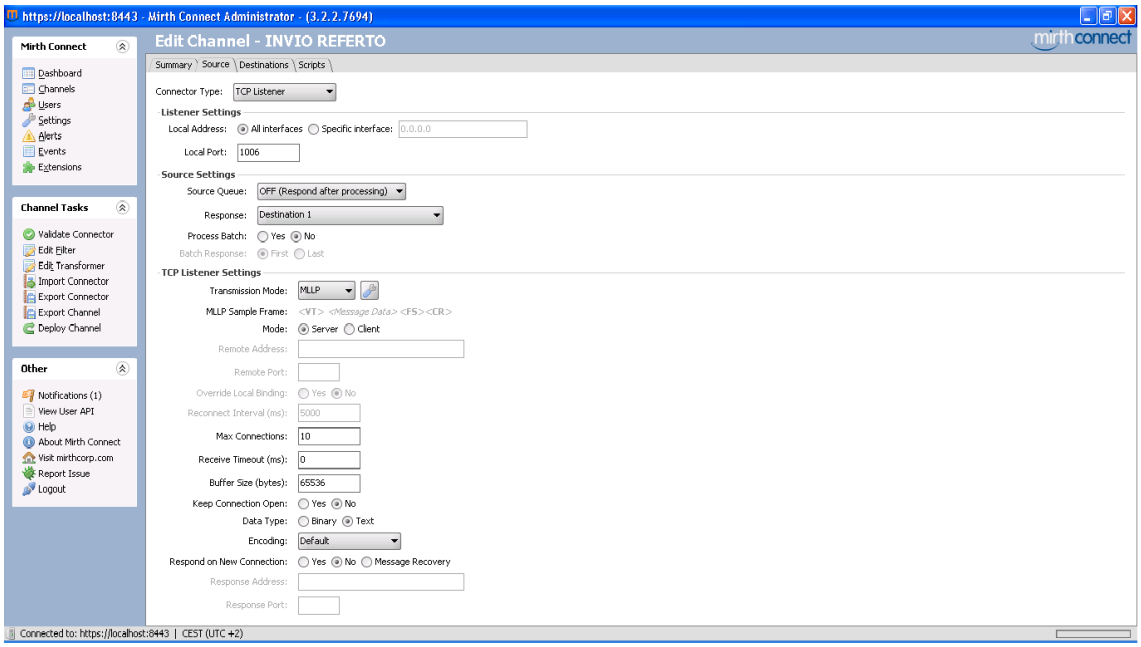

Figura 3.10: Scheda Source dell'Edit Channel

3.3.2 Source

In questa scheda dobbiamo definire i parametri comuni del connettore sorgente e quelli specifici del tipo di connettore.
Iniziamo a vedere quelli comuni. Nella scheda sorgente, a sinistra, compare il pannello **Channel task** con le seguenti funzionalità:

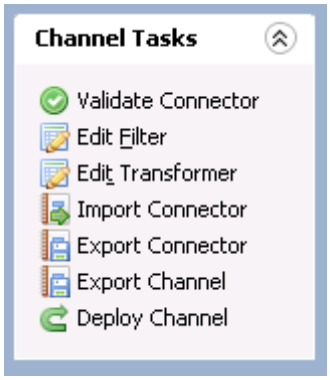

Figura 3.11: Pannello delle funzioni utilizzabili nella scheda Source

- **Validate Connector:** verifica che il connettore sorgente sia corretto
- **Edit filter:** apre la pagina per l'editing del filtro da mettere in sorgente
- **Edit Transformer:** apre la pagina per l'editing del trasformatore da mettere in sorgente
- **Import Connector:** permette l'esportazione del connettore sorgente
- **Export Connector:** permette l'importazione di un connettore sorgente
- **Import Channel:** permette l'esportazione del connettore sorgente
- **Export Channel:** permette l'importazione di un connettore sorgente

Ci sono poi i parametri, comuni ad ogni tipo di connettore, da definire:

- **Connector Type:** appare un menù a tendina in cui possiamo scegliere il tipo di connettore sorgente.

  La scelta del tipo di connettore può essere fatta fra le seguenti possibilità: Channel Reader, DICOM Listener, Database Reader, File Reader, TCP Listener, HTTP Listener, Web Service Listener, JavaScript Reader

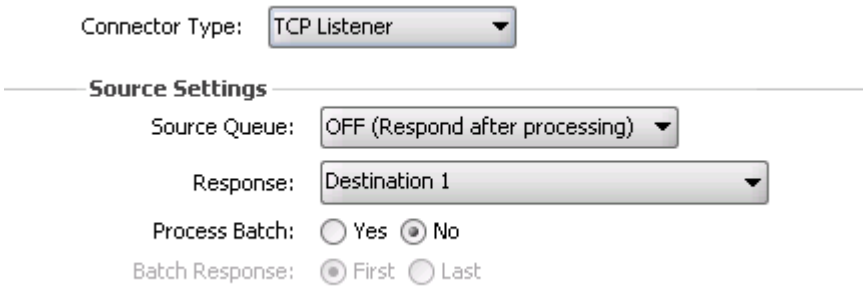

Figura 3.12: Parametri comuni del connettore sorgente

- **Source Queue:** consente di attivare o meno la coda in sorgente.
  - *ON*: appena un messaggio arriva nel connettore sorgente viene messo in coda e viene inviato l'Acknowledgement al sistema sorgente. Solo dopo aver inviato l'Acknowledgment inizia il processing del messaggio. In questo caso non è possibile utilizzare le risposte di Acknowledgement generate dai connettori destinazione. Ma è solo possibile inviare la risposta autogenerata dal Mirth prima del processing oppure non inviarla.
  - *OFF*: viene fatto il processing del messaggio prima di inviare la risposta al sistema sorgente. In questo caso è possibile utilizzare le risposte generate dalle destinazioni.

- **Response:** apre un menù a tendina in cui possiamo scegliere da che punto nel processing del messaggio prelevare la risposta da inviare al sistema sorgente.
  - *None:* Non viene inviata alcuna risposta al sistema sorgente
  - *Auto-generate (Before processing):* Autogenera una risposta prima di processare il messaggio
  - *Auto-generate (After Source Transfomer):* Autogenera una risposta dopo il trasformatore nel connettore sorgente
  - *Auto-generate (Destinations Completed):* Autogenera una risposta una volta completati tutti i connettori destinazione
  - *Postprocessor:* Prende la risposta generata da noi nel Postprocessor Scrip
  - *Destination i:* Prende la risposta generata/ricevuta dalla destinazione i-esima. Per esempio la risposta che arriva nella destinazione dalla Micromed HIS Interface

- **Process Batch:** si può scegliere se abilitare o meno il processing di file batch

- **Batch Response:** ogni messaggio nel file batch viene trattato singolarmente e di conseguenza avrà la sua risposta generata nella maniera selezionata sopra. Dobbiamo qui scegliere se la risposta da inviare al sistema sorgente relativa al file batch sia quella del primo o dell'ultimo messaggio nel file.

TCP listener

In base al tipo di connettore scelto bisogna settare dei parametri specifici. Di seguito vediamo i parametri per il connettore TCP Listener:

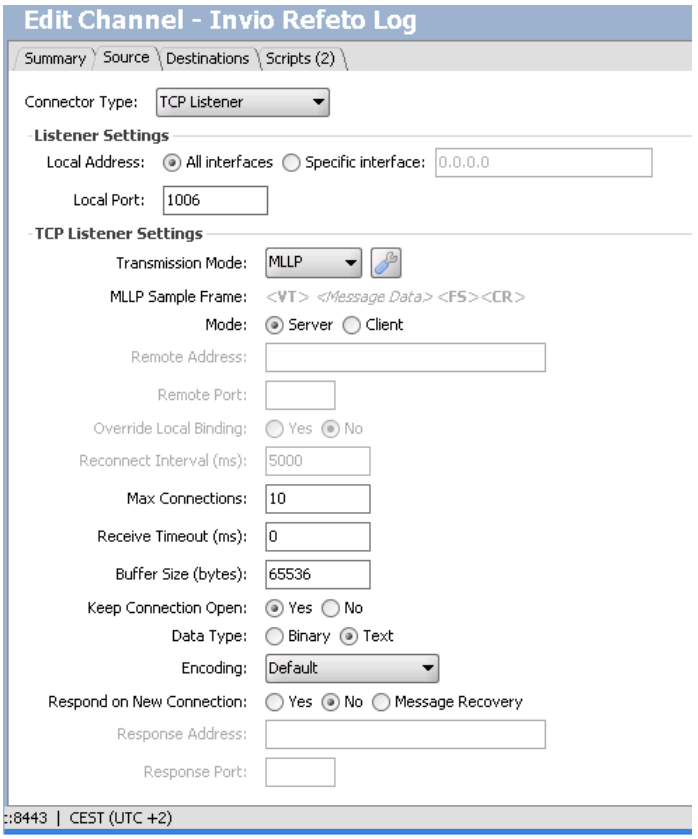

Figura 3.13: Parametri specifici del TCP Listener

- **Local Address:** si può specificare un indirizzo nel quale il connettore sorgente deve mettersi in ascolto
- **Local Port:** inserire la porta alla quale il connettore deve fare riferimento
- **Trasmission Mode:** selezionare il modo i cui vengono trasmessi i messaggi (Basic TCP, MLLP)
- **Mode:** La modalità del connettore.
  - *Server* per ascoltare connessioni dai client
  - *Client* per connettersi ad un TCP server.

    Se si sceglie Client bisogna specificare porte ed indirizzo del server cui bisogna connettersi
- **Max Connection:** numero massimo di connessioni in entrata che il connettore sorgente può avere.
- **Recieve Timeout (ms):** tempo in millisecondi che il connettore deve aspettare senza ricevere messaggi prima di chiudere la connessione. Il valore 0 ms equivale a tempo infinito.
- **Buffer Size (bytes):** scrivere la dimensione in bytes che deve avere il buffer.
- **Keep Connection Open:**
  - *yes:* Il socket rimane aperto finchè il sistema sorgente non chiude la connessione. Selezionando questa opzione sussiste il problema di identificare quando è stato ricevuto un messaggio per poterlo processare. Mirth riconosce l'arrivo di un messaggio in uno dei seguenti casi: il sistema sorgente chiude il socket, vengono letti i caratteri terminatori del messaggio (in modalitá MLLP sono $< FS >< CR >$), scade il Recieve Timeout (ms).
  - *no:* Il socket viene chiuso quando si è finita la ricezione del messaggio
- **Data Type:** selezionare il tipo di messaggi che si ricevono
  - *text:* per flussi di stringhe

  - *binary:* per flussi di dati binari. In questo caso la codifica avviene di default in modalità Base64

- **Encoding:** Scegliere il tipo di codifica da dare al messaggio

- **Respond on New Connection:**

  - *no:* la risposta viene inviata sulla stessa connessione utilizzata per ricevere il messaggio

  - *yes:* la risposta viene sempre inviata su una nuova connessione di cui bisogna settare i parametri

3.3.3 Destinations

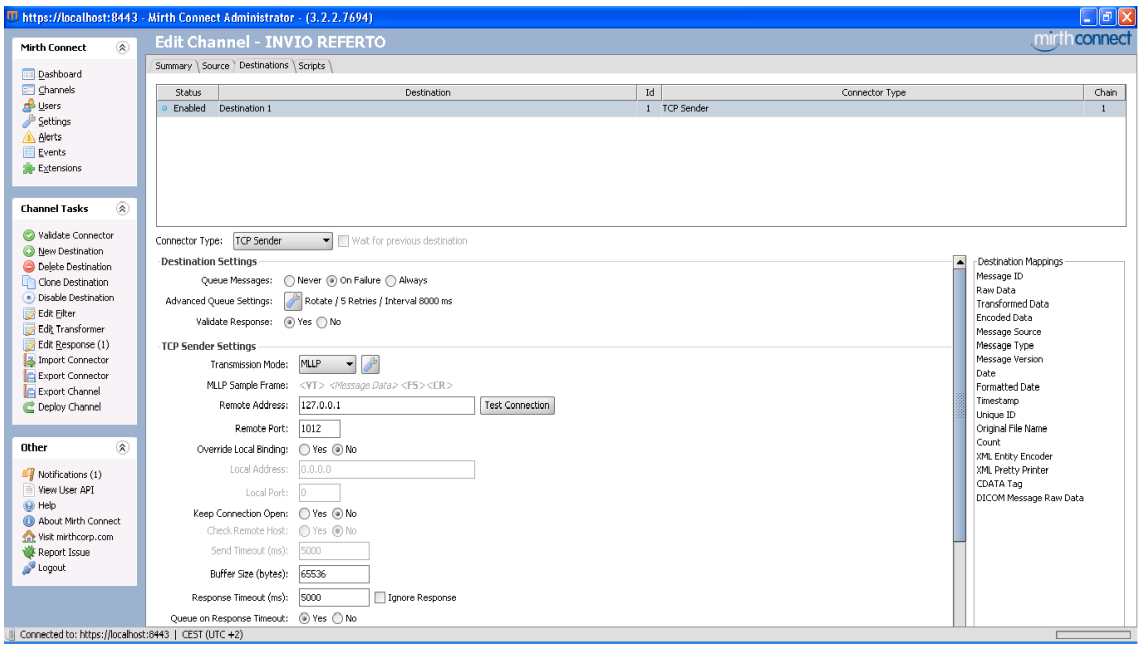

Figura 3.14: Scheda Destinations dell'Edit Channel

In questa scheda dobbiamo definire i parametri comuni dei connettori destinazione e quelli specifici del tipo di connettore. Sulla sinistra il **Channel Tasks** appare come in figura e permette le seguenti funzionalitá:

- **Validate Connector:** verifica che il connettore destinazione sia corretto
- **New Destination:** aggiunge una nuovo connettore destinazione al canale
- **Delete Destination:** elimina il connettore destinazione selezionato dal canale
- **Clone Destination:** clona il connettore destinazione
- **Disable Destination:** disabilita il connettore destinazione selezionato, ciò comporta che nel processing del messaggio quel connettore non viene eseguito

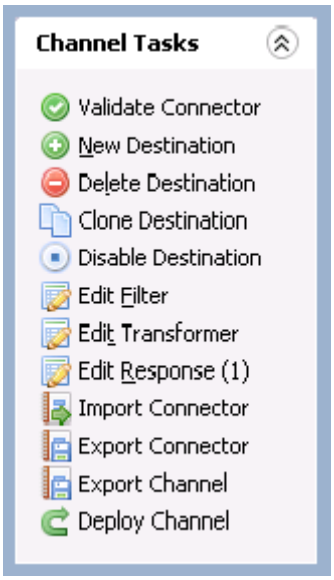

Figura 3.15: Pannello delle funzioni utilizzabili nella scheda Destinations

- **Edit filter:** apre la pagina per l'editing del filtro da mettere in destinazione
- **Edit Transformer:** apre la pagina per l'editing del trasformatore da mettere in destinazione
- **Edit Response:** apre la pagina per l'editing della risposta ricevuta dal sistema informativo in destinazione
- **Import Connector:** permette l'esportazione del connettore destinazione
- **Export Connector:** permette l'importazione di un connettore destinazione
- **Import Channel:** permette l'esportazione del connettore destinazione
- **Export Channel:** permette l'importazione di un connettore destinazione

Ci sono poi i parametri, comuni ad ogni tipo di connettore, da definire:

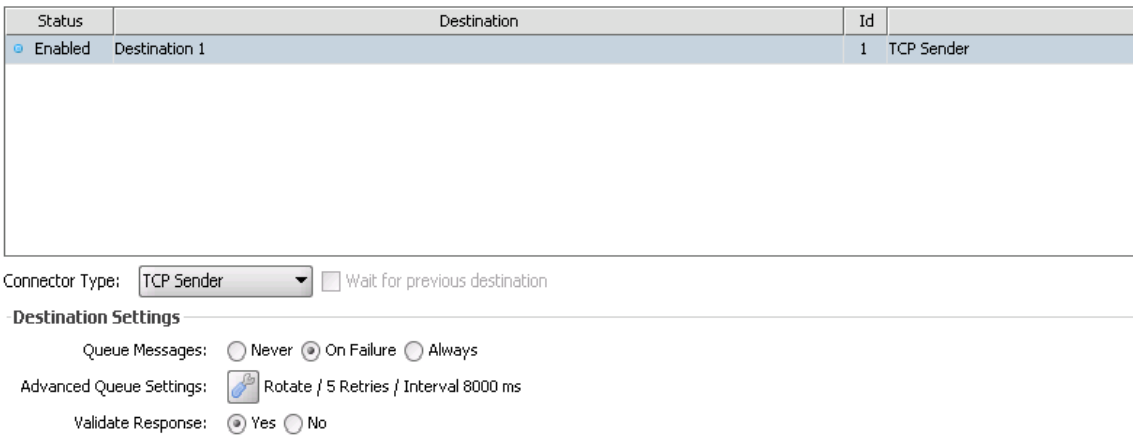

Figura 3.16: Parametri comuni ai vari tipi connettori destinazione

- Nella scheda in alto si possono vedere lo stato, il nome, l'Id ed il tipo, dei connettori destinazione editati.

- **Connector Type:** appare un menú a tendina in cui possiamo scegliere il tipo di connettore destinazione.

  La scelta del tipo di connettore può essere fatta fra le seguenti possibilità: Channel Writer, DICOM Sender, Database Writer, Document Writer, File Writer, TCP Sender, HTTP Sender, Web Service Sender, JavaScript Writer

- **Wait for previous destination:** se abilitato, il connettore destinazione selezionato, prima di essere eseguito, aspetterà che quelli precedenti siano stati completati

- **Queue Message:** si può scegliere quando e se mettere in coda i messaggi

  - *Never:* disabilita la coda in destinazione
  - *On Failure:* prima di mettere in coda il messaggio prova ad inviarlo. Se l'invio fallisce, per problemi di connessione, il messagio viene messo in coda e le destinazioni seguenti ed il Postprocessor vedranno la risposta come QUEUED.
  - *Always:* il messaggio viene immediatamente messo in coda e successivamente si prova l'invio del messaggio. Le destinazioni seguenti ed il Postprocessor vedranno sempre la risposta come QUEUED anche se successivamente l'invio del messaggio andrà a buon fine

- **Advanced Queue Settings:** si possono qui settare i parametri specifici della coda:

  - *Retry Count before Error/Queue:* il numero di volte che il connettore di sorgente deve riprovare ad inviare il messaggio prima di metterlo in coda.
  - *Retry Interval (ms):* intervallo di tempo in millisecondi che il connettore deve far passare fra un tentativo di invio del messaggio e l'altro
  - *Rotate Queue:* abilita la coda circolare. La coda circolare differisce dalla coda normale perchè se non si riesce, per un qualche motivo, ad inviare il messaggio in cima alla coda, quest'ultimo viene spostato in fondo alla coda e si prova ad inviare il successivo. In questo modo un messaggio non blocca la coda evitando l'allungarsi all'infinito della stessa, ma i messaggi vengono consegnati in maniera disordinata, per cui se è importante l'ordine non bisogna abilitare la coda circolare.
  - *Regenerate Template:* ogni volta che il connettore prova ad inviare il messaggio dalla coda vengono rigenerati il template ed altre proprietà del connettore modificando delle variabili
  - *Include Filter/trasformer:* questa opzione è attivabile solo se si è attivato il *Regenerate Template* e, se abilitata, il filtro ed il trasformatore vengono rieseguiti ogni qualvolta si prova l'invio di un messaggio dalla coda
  - *Queue Threads:* il numero di threads che leggono lo coda ed inviano i messaggi simultaneamente. Se l'ordine dei messaggi è importante, il numero di threads non puó essere maggiore di 1. Per esempio, se selezioniamo 2 threads, vengono processati due messaggi simultaneamente.

- **Validate Response:** se abilitato, per ogni risposta che si riceve viene controllato se rispetta dei parametri da noi predefiniti nel *Set Data Types* della scheda **Summary**.

TCP sender

In base al tipo di connettore scelto, bisogna settare dei parametri specifici. Di seguito vediamo i parametri per il connettore TCP Sender:

- **Trasmission Mode:** si seleziona il modo i cui vengono trasmessi i messaggi (Basic TCP, MLLP)
- **Remote Address:** bisogna specificare l'indirizzo IP col quale il connettore sorgente deve mettersi in contatto. È inoltre possibile testare che la connessione esista e funzioni.
- **Remote Port:** bisogna inserire la porta alla quale il connettore deve fare riferimento

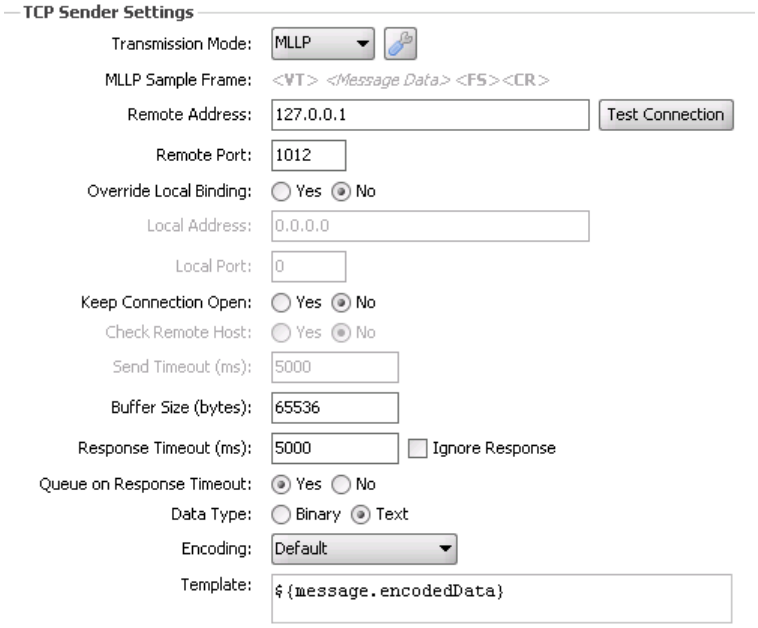

Figura 3.17: Parametri specifici del connettore TCP Sender

- **Keep Connection Open:**
  - *no:* se selezionato, il socket viene chiuso quando si è finito l'invio del messaggio
  - *yes:* se selezionato, il socket rimane aperto durante l'invio di più messaggi. Se si sceglie di mantenere la connessione aperta, si può decidere se controllare, prima dell'invio di un messaggio, se il sistema in destinazione ha chiuso la connessione e bisogna settare il tempo in cui tenere la connessione aperta. Il tempo in cui la connessione deve rimanere aperta può essere un tempo determinato oppure infinito se si inserisce il valore 0.
- **Buffer Size (bytes):** scrivere la dimensione in bytes che deve avere il buffer.
- **Response Timeout (ms):** tempo in millisecondi che il connettore deve aspettare per ricevere l'acknowlwedgment dal sistema in destinazione
- **Ignore Response:** spuntando questa casella il connettore, una volta inviato il messaggio, non si preoccupa di dover ricevere un messaggio di Acknowledgment. In pratica il connettore non si preoccupa se l'invio è andato a buon fine.
- **Queue on Response Timeout:** se abilitato, quando il *Response Timeout* scade, il messaggio viene messo in coda

- **Data Type:** selezionare il tipo di messaggi che si inviano
  - *text:* per flussi di stringhe
  - *binary:* per flussi di dati binari. In questo caso la codifica avviene di default in modalitá Base64
- **Encoding:** Scegliere il tipo di codifica da dare al messaggio
- **Template:** si inserisce il messaggio da inviare. Si può scegliere, ad esempio, fra le varie tipologie di messaggio che sono state generate nel corso del processing (raw, transformed, encoded), che sono messe a disposizione nella lista a destra, la *Destination Mapping*.

Si noti che i messaggi vengono messi in coda solo se scade il timeout della risposta o ci sono problemi di connessione, nel caso in cui sia stata selezionata l'opzione *on failure* relativa alla coda in destinazione nel riquadro **Destination Settings** e abilitata l'opzione *queue on response timeout* nel riquadro **TCP Sender Settings**. Nel caso in cui il messaggio riceva un NACK il messaggio viene considerato in Errore.

File writer

- **Method:** selezionare il metodo utilizzato per accedere ai file da scrivere
- **Directory:** il percorso della cartella in cui è contenuto il file da scrivere
- **File Name:** il nome del file da scrivere

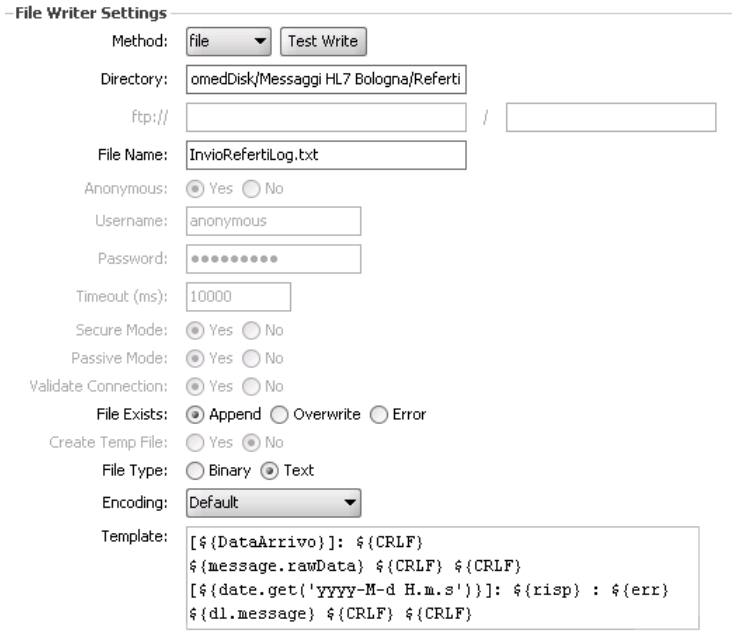

Figura 3.18: Parametri specifici del connettore File Writer

- **File Exist:** Bisogna selezionare cosa fare nel caso il file esista giá:
  - *append:* in questo caso si va ad aggiungere al file esistente ciò che bisogna scrivere
  - *Overwrite:* Il file viene sostituito completamente con ciò che andiamo a scrivere di nuovo perdendo tutti i dati precedenti contenuti nel file
  - *Error:* in questo caso se il file col nome specificato già esiste il messaggio verrà considerato in errore
- **File Type:** selezionare il tipo di messaggi che si inviano

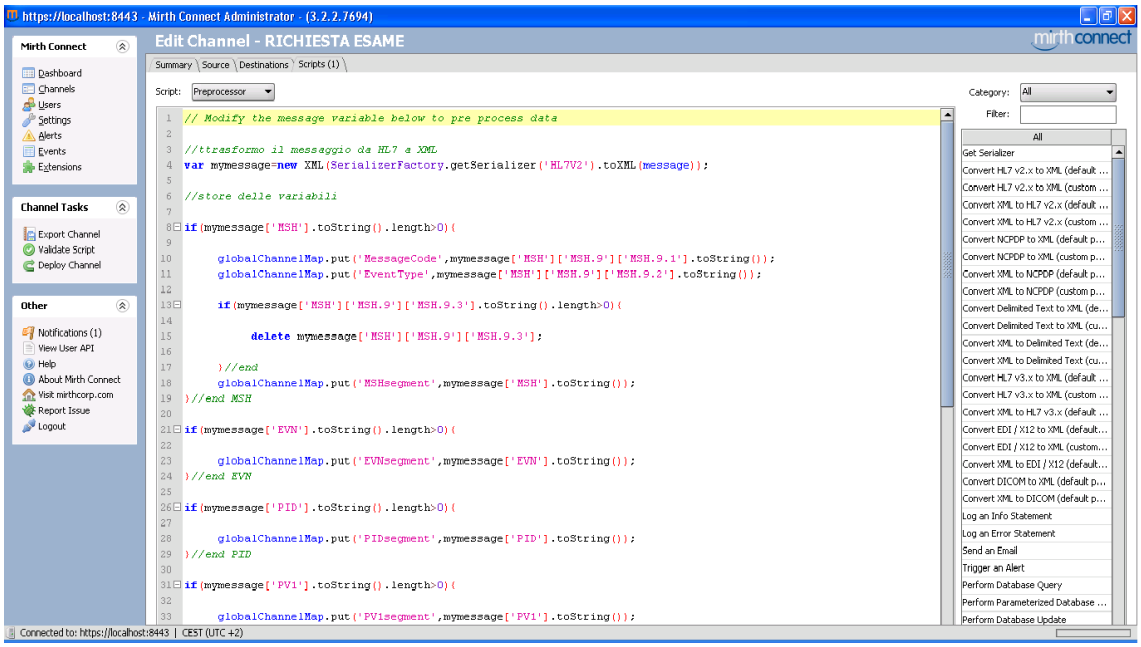

Figura 3.19: Scheda Scripts dell'Edit Channel

  - *text:* per flussi di stringhe
  - *binary:* per flussi di dati binari. In questo caso la codifica avviene di default in modalità Base64

- **Encoding:** Scegliere il tipo di codifica da dare al messaggio
- **Template:** il template del messaggio da inviare

3.3.4 Scripts

In questa scheda vengono messi a disposizione 4 script da editare. Ognuno di questi script viene eseguito in momenti diversi nel workflow del canale.

- **Deploy:** : Lo script viene eseguito solo quando il canale viene avviato. Solitamente è utilizzato per definire variabili globali del canale o caricare dati.
- **Preprocessor:** Lo script viene eseguito prima che il messaggio in arrivo venga inviato all'encoder. Spesso utilizzato per rimuovere caratteri invalidi o sistemare campi invalidi del messaggio.
- **Postprocessor** Lo script viene eseguito dopo che tutte le destinazioni sono state elaborate. Spesso utilizzati per creare ACKs/NACKs da inviare in risposta.
- **Undeploy:** Lo script viene eseguito quando il canale viene chiuso. Generalmente utilizzati per pulire le mappe e chiudere le connessioni.

3.3.5 Edit Filter

Sia nel connettore Sorgente, che in ogni connettore Destinazione è possibile inserire un filtro. Ogni filtro è composto da una serie di *Regole* che vengono valutate come

un'espressione booleana. Ogni regola ha come risultato un valore booleano, il valore finale viene valutato come la concatenazione logica dei risultati di ogni singola regola. Se il risultato del filtro è false il messaggio viene filtrato, altrimenti procede nel workflow.
Per accedere all'editing del filtro bisogna selezionare l'*Edit Filter* nel connettore in cui si vuole inserirlo, compare così la schermata in figura, dove possiamo costruire le regole del filtro. La variabile diponibile con la quale costruire le regole è **msg**.

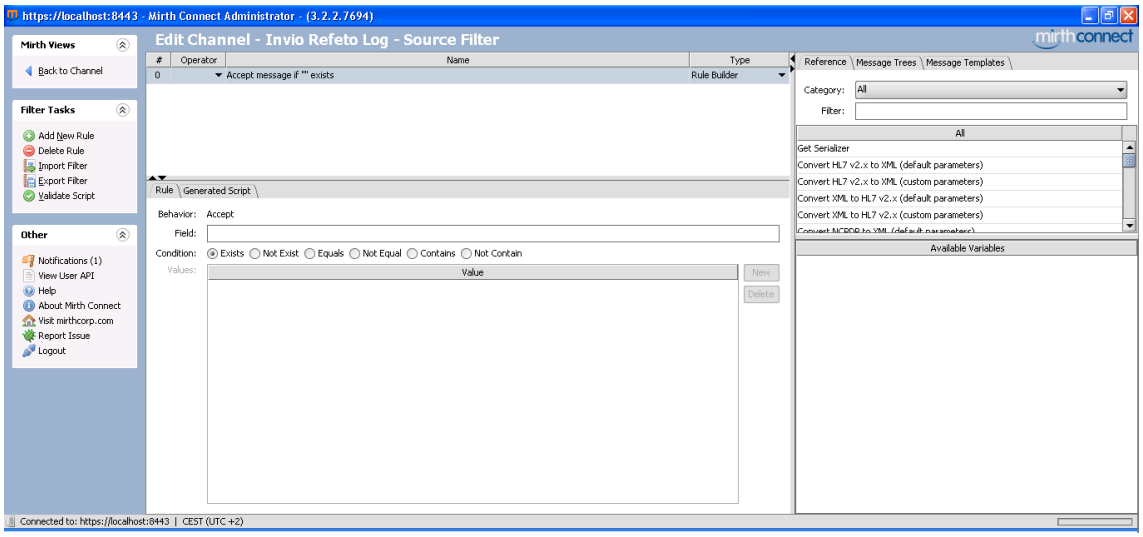

Figura 3.20: Schermata per l'editing di un filtro

Analizziamo innanzitutto il pannello delle **Filter Tasks**:

- **Add New Rule:** aggiunge una nuova regola
- **Delete Rule:** elimina la regola selezionata
- **Import Filter:** permette l'importazione di un filtro, in formato XML
- **Export Filter:** permette l'esportazione del filtro, in formato XML
- **Validate Script:** valida gli script generati nelle regole
- **Move Rule Up/Down:** disponibile solo se le regole impostate nel filtro sono più di una. Permette di spostare la regola più in alto, o in basso, in base al fatto che vogliamo venga eseguita prima, o dopo, un'altra regola.

La generazione di un filtro passa per l'editing di ogni singola regola. Innanzitutto, bisogna aggiungere una regola al filtro cliccando su **Add New Rule**. Nella sezione *Type* possiamo decidere il tipo di regola fra:

- **JavaScript:** mette a disposizione uno script in cui possiamo generare manualmente il nostro codice
- **External Script:** richiede il percorso da cui caricare il codice generato esternamente
- **Rule Builder:** permette l'editing di semplici regole mettendo a disposizione dei codici precompilati in cui dobbiamo inserire solo le variabili. Vengono messi a disposizione i codici per le seguenti condizioni:

- *Exists:* controlla se un determinato segmento o campo è presente nel messaggio
- *Not Exist:* controlla se un determinato segmento o campo non è presente nel messaggio
- *Equals:* controlla se un determinato campo in un segmento è uguale ad una variabile (stringa o numero) da noi definita
- *Not Equals:* controlla se un determinato campo in un segmento è diverso ad una variabile (stringa o numero) da noi definita
- *Contain:* controlla se il campo di un segmento contiene un determinato valore
- *Not Contain:* controlla se il campo di un segmento non contiene un determinato valore

Nella sezione a sinistra sono diponibili tre schede:

- Reference: Vengono visualizzate le function messe a disposizione da Mirth per l'editing del filtro. È possibile sia visualizzarle per raggruppamenti, sia cercare una determinata funzione mediante delle operazioni di filtraggio
- Message Template: in questa scheda si può inserire il template dei messaggi che tipicamente vengono processati dal canale
- Message Tree: viene mostrata, mediante una struttura ad albero, la rappresentazione XML del messaggio inserito nel Message Template. Da questa scheda è possibile portare lo specifico campo del messaggio nello script con tecnica *drag and drop*.

3.3.6 Edit Transformer

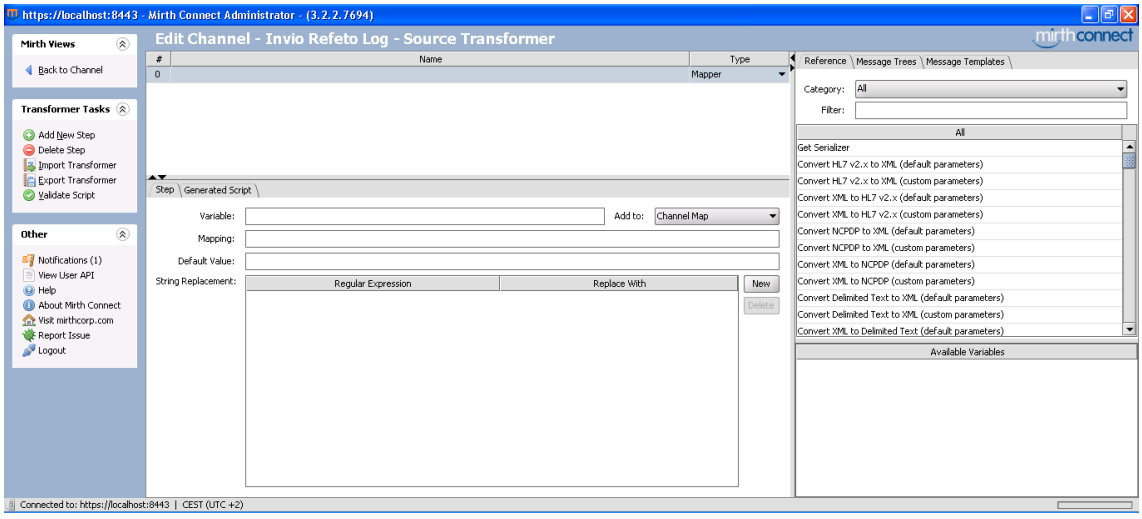

Figura 3.21: Schermata per l'editing di un trasformatore

Sia nel connettore Sorgente che in ogni connettore Destinazione è possibile inserire un trasformatore. Ogni trasformatore viene costituito da dei passi che vengono eseguiti in ordine. Lo scopo di ciascuno di questi passi può essere differente, per

esempio si può caricare una variabile in una mappa o costruire il messaggio in uscita al trasformatore. Il trasformatore mette a disposizione due variabili su cui eseguire le operazioni impostate nei passi, **msg**, relativa al messaggio in entrata al trasformatore, e **tmp**, relativa al messaggio in uscita.
Per accedere all'editing del trasformatore bisogna selezionare l'*Edit Transformer* nel connettore in cui si vuole inserirlo, compare così la schermata in figura, dove possiamo costruire i passi del trasformatore.
Analizziamo il pannello delle **Transfomer Tasks**:

- **Add New Step:** aggiunge un nuovo passo
- **Delete Step:** elimina il passo selezionato
- **Import Transformer:** permette l'importazione di un trasformatore
- **Export Transformer:** permette l'esportazione del trasformatore
- **Validate Script:** valida gli script generati nei passi
- **Move Step Up/Down:** disponibile solo se i passi impostati nel trasformatore sono più di uno. Permette di spostare il passo più in alto, o in basso, in base al fatto che vogliamo venga eseguito prima, o dopo, un altro passo.

La generazione di un trasformatore passa per l'editing di ogni singolo passo. Innanzitutto bisogna aggiungere un passo al filtro cliccando su **Add New Step**. Nella sezione *Type* possiamo decidere il tipo di passo fra:

- **JavaScript:** mette a disposizione uno script in cui possiamo generare manualmente il nostro codice
- **External Script:** richiede il percorso da cui caricare il codice generato esternamente
- **Mapper:** permette il salvataggio delle variabili nelle mappe. Bisogna definire
  - *Variable:* il nome con cui la variabile dev'essere salvata nella mappa
  - *Map:* la mappa in cui vogliamo venga salvata la variabile
  - *Add To:* il percorso del segmento, campo, sottocampo, che vogliamo andare a prelevare dal **msg**
  - *Default Value:* un valore predefinito che vogliamo andare a salvare nella mappa
- **Message Builder:** permette l'editing dei passi mettendo a disposizione dei codici precompilati in cui dobbiamo inserire solo le variabili. In particolare dobbiamo inserire:
  - *Message Segment:* il percorso del segmento, campo, sottocampo, che vogliamo andare a scrivere nella variabile **tmp**
  - *Mapping:* il percorso del segmento, campo, sottocampo, che vogliamo andare a prelevare dalla variabile **msg**
  - *Default Value:* un valore predefinito che vogliamo andare a scrivere nella variabile **tmp**

Nella sezione a sinistra sono disponibili tre schede:

- **Reference:** Vengono visualizzate le function messe a disposizione da Mirth per l'editing del trasformatore. È possibile sia visualizzarle per raggruppamenti sia cercare una determinata funzione mediante delle operazioni di filtraggio

- **Message Template:** in questa scheda si possono inserire i template dei messaggi in ingresso (Inbound) ed in uscita (Outbound) dal trasformatore che tipicamente vengono processati dal canale

- **Message Tree:** vengono mostrate, mediante una struttura ad albero, le rappresentazioni XML dei messaggi in entrata ed in uscita dal trasformatore inseriti nel Message Template. Da questa scheda è possibile portare lo specifico campo del messaggio nello script con tecnica *drag and drop*.

3.3.7 Edit Response

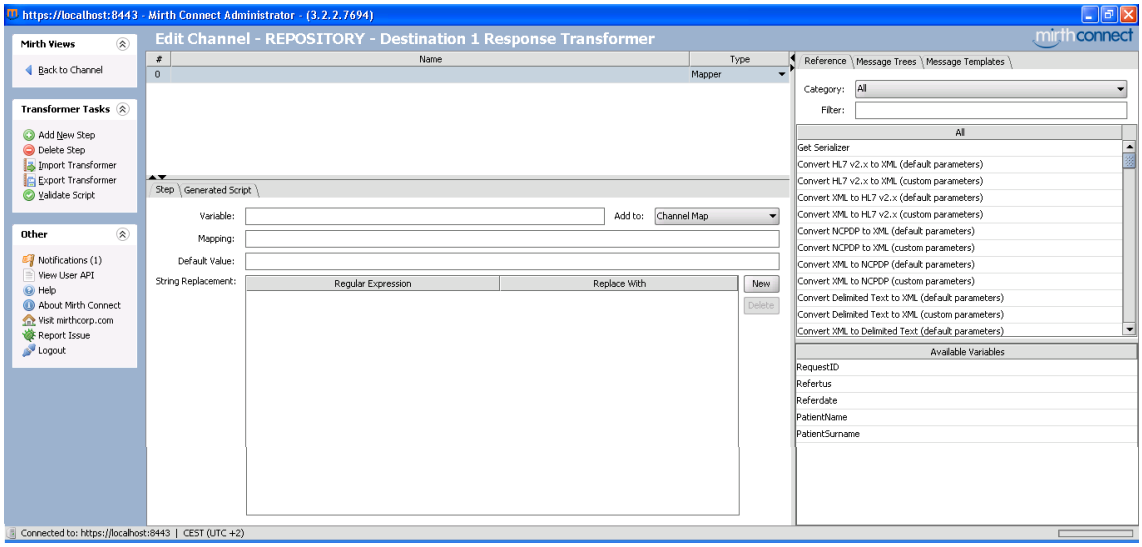

Figura 3.22: Schermata per l'editing di un trasformatore per i messaggi di Acknowledgment

Solo nei connettori Destinazione è possibile inserire un trasformatore che ha le stesse funzionalità del trasformatore normale, ma l'Inbound message, **msg**, è l'acknowledgement ricevuto dal sistema in destinazione, l'Outbound message, **tmp**, è l'acknowledgment modificato secondo le nostre esigenze che poi verrà inviato al sistema sorgente.
La costruzione del messaggio in uscita al trasformatore avviene sempre tramite dei passi che vengono eseguiti in ordine e che vengono generati allo stesso modo che nell'*Edit Transformer*. Per scrivere il trasformatore per gli Acknowledgment bisogna selezionare l'*Edit Response* nel connettore destinazione in cui si vuole scriverlo, compare così la schermata in figura, dove possiamo costruire i passi del trasformatore.

L'*Edit Response* viene eseguito solo a fronte di una risposta ricevuta dal sistema in destinazione. Per esempio, se non viene ricevuta alcuna risposta perchè il tempo di ricezione scade, lo script non viene eseguito.

3.4 Monitoraggio del canale

3.4.1 Channel Messages

Nella ***Dashboard*** selezionando **View Message**, o cliccando due volte col tasto sinistro sul canale, si accede alla schermata **Channel Messages**.

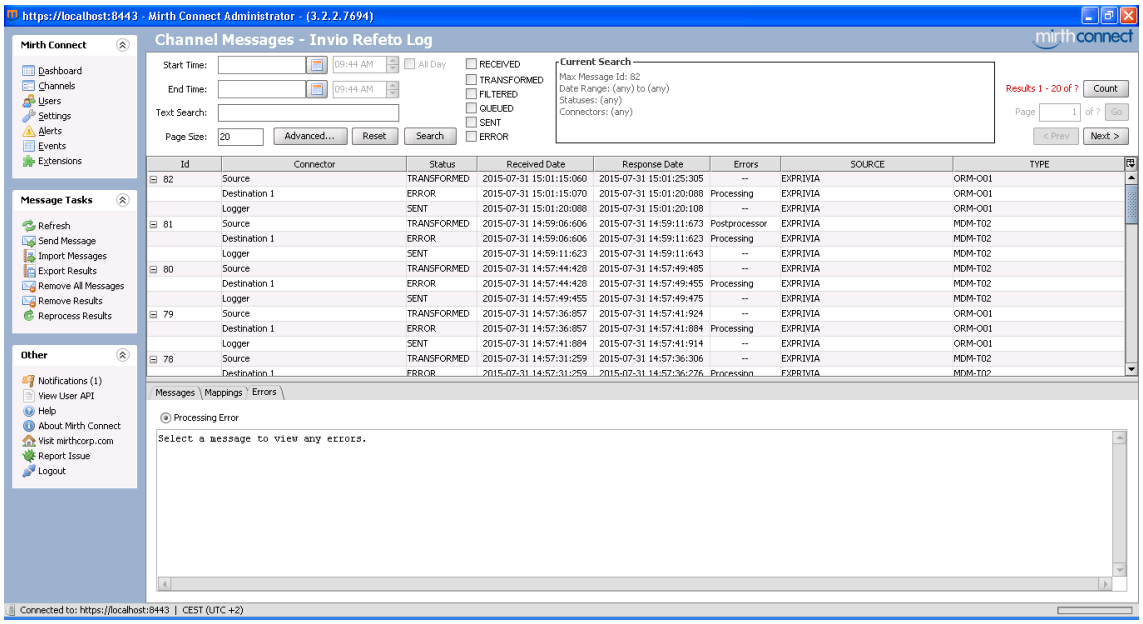

Figura 3.23: Schermata Channel Messages

In questa schermata possiamo vedere lo storico dei messaggi processati dal canale, o dal singolo connettore, vedere il messaggio in sue diverse versioni (Raw, Transformed, Response etc...), i Custom Metadata, lo stato finale che il processing del messaggio ha generato, in che punto del workflow del canale si sono verificati gli errori e che tipo di errori si sono verificati.
Alla sinistra appare il pannello **Message task** che mette a disposizione le seguenti funzioni:

- **Refresh:** aggiorna i messaggi processati dal canale o dal connettore
- **Send Message:** permette l'invio di un messaggio nel canale, o connettore, selezionato
- **Import Messages:** permette l'importazione di messaggi da un file
- **Export Results:** esporta i messaggi risultato della ricerca fatta. I messaggi vengono salvati, in formato XML, ciascuno in un file separato nella cartella selezionata
- **Remove All Messages:** Rimuove tutti i messaggi del canale selezionato, azzerandone le statistiche

Figura 3.24: Pannello delle funzioni utilizzabbili nella Channel Messagges

- **Remove Results:** rimuove i messaggi risultato della ricerca effettuata

- **Remove Message:** rimuove il singolo messaggio
- **Reprocess Results:** permette il reprocessing dei messaggi risultato della ricerca effettuata successivamente é possibile selezionare quali connettori destinazione devono riprocessare il messaggio.
- **Reprocess Message:** permette il reprocessing del singolo messaggio (è possibile selezionare quali connettori destinazione devono riprocessare il messaggio).

Capitolo 4

Implementazioni sul campo

4.1 Training e test svolti in laboratorio

La prima fase del mio periodo di stage alla Micromed S.p.A. è stata centrata sulla comprensione, almeno in parte, delle molteplici potenzialità di Mirth Connect. In particolare, ho inizialmente approfondito uno studio teorico del programma utilizzando il materiale messomi a disposizione dalla rete. Successivamente sono passato alla fase pratica per imparare ad utilizzare il programma.
In questa fase ho lavorato su una macchina in cui, oltre ad aver installato Mirth Connect, erano installate anche la Micromed HIS Interface, System Plus Evolution e HL7 Sender, un programma sviluppato da Micromed per simulare il CUP di un SIO. Per poter effettuare dei test completi era necessario poter simulare anche il Repository: di conseguenza, una volta effettuati i primi test di connessione, sono state create due istanze di Mirth per simulare un SIO in modo da poter riprodurre situazioni più realistiche. Le istanze del Mirth sono a tutti gli effetti dei canali Mirth che però vengono utilizzati per simulare, ad esempio, il comportamento del Repository e del CUP, invece che per consegnare, trasformare o filtrare i messaggi. Sono state, dunque, sviluppate due istanze che simulassero il SIO:

- il Centro Unico Prenotazioni per simulare l'invio delle richieste a System Plus Evolution, che ha sostituito l'HL7 Sender
- il deposito referti per simulare la ricezione dei referti generati da System Plus Evolution

Una volta implementato l'HIS simulato sono stati creati due canali Mirth: uno che si occupasse dell'invio delle richeste esame ed uno per l'invio dei referti. In figura 4.1 è rappresentata l'interfaccia di test che è stata sviluppata per risolvere le problematiche riscontrate dall'azienda e per lo sviluppo dei canali che successivamente sono stati utilizzati nelle interfacce degli ospedali di Bologna e Milano.

Di seguito sono riportate alcune delle problematiche che si sono dovute risolvere prima di pensare all'implementazione in un ospedale dell'interfaccia.

4.1.1 Gestione ACK in risposta ad una Richiesta

Il primo problema riscontrato nell'inserimento del Mirth fra l'HIS simulato, con l'HL7 Sender, e la Micromed HIS Interface è stato la gestione del messaggio di acknowledgement.

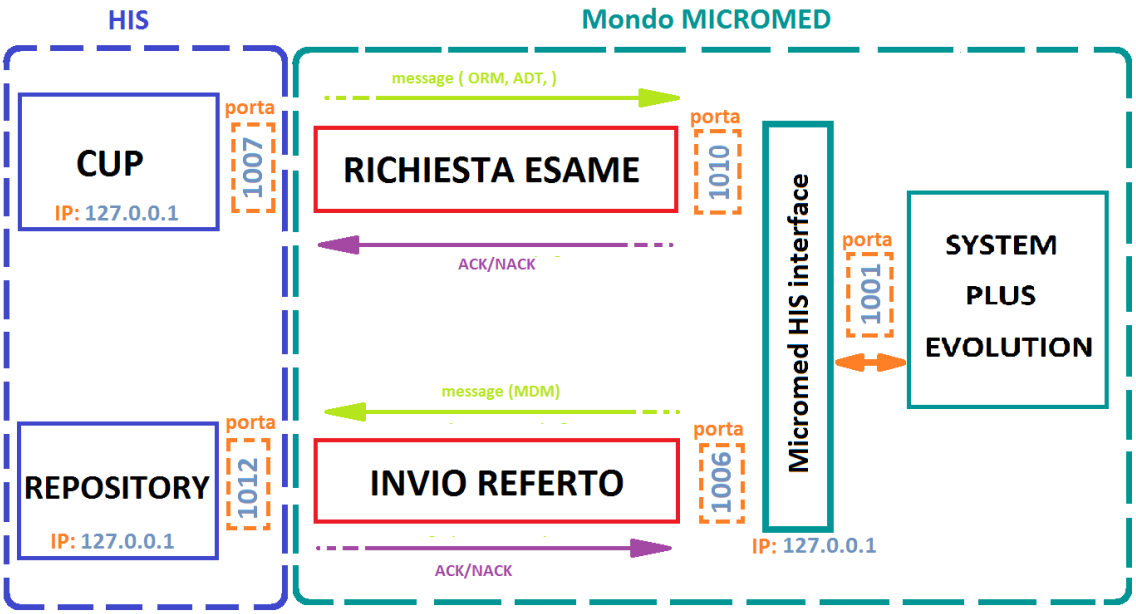

Figura 4.1: Interfaccia di test fra HIS simulato e System Plus Evolution utilizzando Mirth

In particolare, una volta collegato Mirth ai due dispositivi, il messaggio inviato dall'HIS verso la Micromed HIS Interface veniva correttamente consegnato, ma non veniva riconsegnato l'ACK generato dalla Micromed HIS Interface.

La soluzione a tale problema 'e stata utilizzare L'*Edit Response* nel connettore destinazione. Soluzione che ha messo in evidenza un importante potenzialità di Mirth Connect. Prendere confidenza con l'*Edit Response* e comprendere il meccanismo di selezione della risposta è stato essenziale per poter sviluppare in maniera efficiente le interfacce che sono state implementate nei Sistemi Informativi Sanitari Ospedalieri del Sant'Orsola di Bologna e del San Raffaele di Milano.

L'*Edit Response* nel connettore destinazione del canale Mirth è stato configurato in modo da prelevare la risposta ricevuta dal sistema in destinazione e salvarla. Per far si che questa fosse la risposta che effettivamente veniva inviata all'HIS si è dovuto selezionare, nella scheda Source, la destinazione da cui prelevare la response.

Nel caso in cui nel connettore destinazione sia stato deciso di mettere in coda i messaggi, oppure non sia stata ricevuta alcuna risposta dal sistema in destinazione, si verificano degli errori. L'errore sta nel fatto che in questi due casi il canale Mirth autogenera una risposta con la quale identifica lo stato del messaggio, QUEUED o ERRORED rispettivamente, ma tale risposta autogenerata è una risposta vuota, non contiene cioè alcun messaggio di Acknowledgment. Dipenderà quindi dalle specifiche del sistema se tale risposta verrà accettata come valida.

In conclusione, per risolvere il problema della corretta consegna del ACK/NACK bisogna:

- Selezionare, nel connettore sorgente, da dove prelevare la risposta

- Nel connettore destinazione, utilizzando l'*Edit Response* generare la risposta che si desidera.

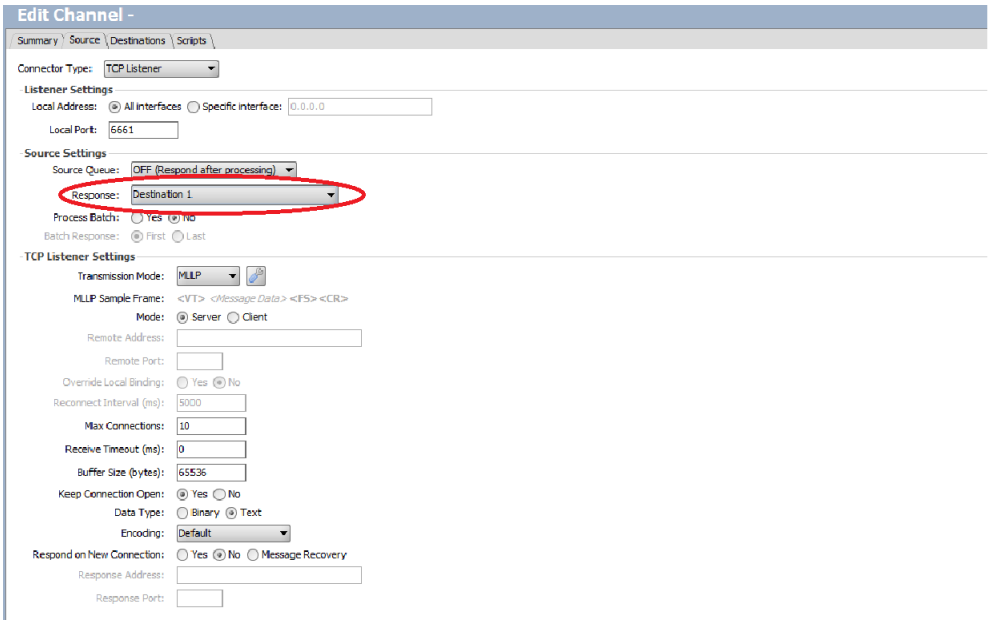

Figura 4.2: Parametri del connettore Sorgente TCP Listener. In rosso viene evidenziata la scelta della risposta

4.1.2 Filtraggio

Qualora mancasse a monte un filtro sui messaggi che arrivano alla Micromed HIS Interface è possibile utilizzare Mirth per filtrare i messaggi in entrata in base ad alcuni criteri.
In particolare è stato creato un filtro nel connettore destinazione utilizzando l'*Edit filter*, e si è scelto di utilizzare un Javascript per filtrare i messaggi in base alle seguenti esigenze:

1. Vengono accettati solo i messaggi di interesse per SystemPlus Evolution, quali, ad esempio: ORM^O01, ADT^A01, ADT^A08, ADT^A31, ADT^A40. Se il messaggio arrivato è diverso da quelli elencati viene filtrato.

2. Su questi messaggi viene fatto un controllo sulla presenza di tutti i segmenti necessari per la corretta esecuzione della richiesta. Per esempio il messaggio ADT^A40 deve contenere obbligatoriamente i segmenti: MSH, EVN, PID, PV1, MRG. Se manca uno di questi segmenti il messaggio viene filtrato, e viene inviato un Acknowledgment al sistema che l'ha spedito per informarli che al messaggio manca un segmento.

Nello scrivere il codice del filtro bisogna porre attenzione ai seguenti punti:

- Nell'*Edit Filter* l'Inbound message è in formato XML quindi per accedere ai campi d'interesse nei vari segmenti bisogna utilizzare una sintassi del tipo:

```
1 msg['MSH']['MSH.1']['MSH.1.1'];
```

  e per accedere alla variabile contenuta nel campo desiderato si utilizza il comando .toString()

- Per controllare che i segmenti siano presenti non si può utilizzare la seguente sintassi:

```
msg['MSH']!=null;
```

ma bisogna fare un controllo sulla lunghezza del segmento MSH. Intuitivamente se è maggiore di zero il segmento è presente, quindi è stato utilizzato:

```
msg['MSH'].toString().length()>0;
```

- Ogni Javascript o Rule contenuta nei diversi passi del filtro deve ritornare un valore booleano, true o false. In ogni passo si può scegliere un operatore logico, AND o OR. Quindi ogni passo ha il suo valore booleano ed operatore logico. Per decidere se il messaggio viene filtrato o meno l'*Edit Filter* valuta la concatenazione logica dei risultati dei passi del filtro.

  Se il risultato finale sarà false il messaggio verrà filtrato altrimenti passa al trasformatore.

4.1.3 Trasformazione

Una volta superata la fase di filtraggio si passa alla fase di trasformazione del messaggio. La possibilità di trasformare il messaggio viene data nell'*Edit Transformer*. In questo editor bisogna fare riferimento a due messaggi:

- Inbound Message: il messaggio ***da trasformare*** a cui si ha accesso nel formato XML tramite la variabile **msg**
- Outbound Message: il messaggio ***trasformato*** a cui si ha accesso nel formato XML tramite la variabile **tmp**.

A livello pratico si è cercato di risolvere alcune delle problematiche che sono state riscontrate in certe realtà ospedaliere:

1. Trasformazione di un messaggio HL7 di tipo ADT^A01 in uno di tipo ORM ^O01.
2. Trasformazione di un messaggio HL7 di tipo ADT^A08 in uno di tipo ADT ^A31.
3. Trasformazione del segmento MSH in modo tale che il campo adibito all'identificazione del tipo di messaggio non contenga componenti superflue o ridondanti.
4. Trasformazione nulla del messaggio.

Di seguito vediamo come sono state risolte queste problematiche:

1. Alcuni ospedali, soprattutto americani, gestiscono la richiesta esame, che dovrebbe essere gestita con un ORM^O01, tramite il messaggio ADT^A01 che è un tipo di messaggio adibito all'inserimento di un nuovo paziente nel database. Poichè System Plus gestisce gli ORM per le richieste esame è necessario trasformare i messaggi da ADT^A01 a ORM^O01.

   I due messaggi sono di tipo diverso ma contengono le stesse informazioni anche se in campi differenti; è allora necessario spostare le informazioni nei campi giusti ed aggiungere o togliere i segmenti necessari o non necessari.

Per esempio l'ADT^A01 contiene i segmenti MSH, EVN, PID, PV1. L'ORM ^O01 invece i segmenti MSH, PID, PV1, OBR, ORC. Per trasformare il messaggio bisogna aggiungere i segmenti OBR e ORC ed eliminare il segmento EVN, tali operazioni si fanno con le function **delete** e **createSegment()**.

Nel segmento ORC dell'ORM^O01 vanno inserite le informazioni sulla richiesta d'esame in alcuni campi specifici quali: in ORC.2.1 il codice identificativo della richiesta, in ORC.9.1 l'orario della richiesta, in ORC.12.1 il nome del dottore richiedente.

ATTENZIONE: l'eliminazione o aggiunta dei segmenti nel messaggio va fatta in funzione dell'*Outbound message template* inserito. Se, come nel nostro caso, il template del messaggio in uscita ha i segmenti MSH, EVN, PID, PV1 vanno fatte le operazioni che abbiamo descritto sopra.

Di seguito viene presentato un esempio di codice per questo tipo di trasformazione. Le variabili a cui si accede nella GlobalChannel Map sono state definite nel *preprocessor*.

```
if($gc('EventType')=='A01') {

                //Definizione dei segmenti per l'ORM^O01
                tmp['MSH']=msg['MSH'];
                delete tmp['EVN'];
                tmp['PID']=msg['PID'];
                tmp['PV1']=msg['PV1'];
                createSegment('OBR', tmp)
                createSegment('ORC', tmp)
                tmp['ORC']['ORC.1']['ORC.1.1']='NW';
                //Identificativo Richiesta
                tmp['ORC']['ORC.2']['ORC.2.1']=$gc('PV1RequestID');
                //Ora della richiesta
                tmp['ORC']['ORC.9']['ORC.9.1']=$gc('PV1RequestTime');
                //Nome dott richiedente l'esame
                tmp['ORC']['ORC.12']['ORC.12.1']=$gc('PV1DoctorName');

                //campo riempito a caso nel segmento OBR
                tmp['OBR']['OBR.1']['OBR.1.1']='1';

                //sostituzione messageCode EventType
                tmp['MSH']['MSH.9']['MSH.9.1']='ORM';
                tmp['MSH']['MSH.9']['MSH.9.2']='O01';

        }// end event A01
```

2. gli eventi A08 ed A31, riguardanti l'ADT, identificano l'aggiornamento dei dati, rispettivamente, del paziente e della persona. È facile intuire che le informazioni contenute nei due messaggi sono le stesse, quindi la trasformazione da ADT^A08 ad ADT^A31 comporta una semplice riassegnazione del codice identificativo dell'evento nel campo MSH.9.2 e in EVN.1.1.

   Di seguito un esempio di codice:

```
if($gc('EventType')=='A08') {
        //Define MSH segment
```

```
        tmp['MSH']=msg['MSH'];
        //Define EVN segment
        tmp['EVN']=msg['EVN'];
        //Define PID
        tmp['PID']=msg['PID'];

        //Replace EventType
        tmp['MSH']['MSH.9']['MSH.9.2']='A31';
        //Replace Event Code
        tmp['EVN']['EVN.1']['EVN.1.1']='A31';
        delete tmp['PV1']

}// end event
```

3. Può succedere che alcuni ospedali inseriscano un componente ridondante nel campo MSH.9 che identifica il tipo di messaggio. Per esempio: ADT^A31^ADT_A31. In questo caso può risultare utile eliminare la parte ADT_A31 che è la terza componente del campo MSH.9 . Un esempio di come fare per eliminarla è:

```
if(mymessage['MSH']['MSH.9']['MSH.9.3'].toString().length>0){

        delete mymessage['MSH']['MSH.9']['MSH.9.3'];

}//end
```

4. Può succedere che il messaggio in entrata al trasformatore non debba subire alcuna trasformazione, per esempio se arriva un ORM^O01. Per risolvere questa esigenza è stato aggiunto un nuovo passo nell'*Edit transformer* in modo che riconosca questo tipo di messaggio e lo lasci passare indenne per il trasformatore applicando una semplice assegnazione;

```
if(msg['MSH']['MSH.9']['MSH.9.1'].toString()=='ORM'){

        tmp=msg;
}
```

4.1.4 HIS simulato

Per poter effettuare ulteriori test è stato utilizzato Mirth per creare dei canali che simulassero il comportamento di un HIS. Le esigenze per cui è stato creato l'HIS simulato, composto dalle due istanze di Mirth, sono principalmente due:

- test sul canale per l'invio referti
- generazione di una grossa mole di messaggi da inviare a System Plus Evolution per controllare come si comporta il canale di collegamento sotto stress

4.1.5 Ricezione del referto

La gestione e la responsabilità delle code per i messaggi in uscita, referti, da System Plus Evolution è un punto critico che può essere gestito da Mirth. Le situazioni in cui ci si può trovare sono principalmente 3:

1. arriva una risposta positiva: tutto ok
2. arriva nessuna risposta: bisogna reinviare il referto
3. arriva una risposta negativa: bisogna per lo meno generare un alert

Come prima cosa è stato inserito un canale di ritorno per i referti fra la Micromed HIS Interface ed HL7 sender, applicativo che simulava l'HIS. Il canale è stato impostato in modo da poter risolvere i casi elencati sopra:

1. Il primo problema che è stato riscontrato è che l'HL7 sender non generava nessun messaggio di acknowledgment una volta ricevuto il referto. Per questo motivo è stato progettato un canale in Mirth che simulasse il Repository al posto dell'HL7 sender, in maniera tale che fosse previsto un acknowledgment da parte dell'HIS per evitare il reinvio continuo del messaggio da parte del canale di trasferimento referti. Per far ciò il connettore sorgente del canale che simula l'HIS è stato impostato in modo che prima di processare i messaggi li metta in coda ed invii un ACK di avvenuta ricezione.

2. Per risolvere questa esigenza sono stati scelti dei parametri specifici nel connettore sorgente e destinazione del canale di trasporto referti.

   In particolare si è scelto di non avere la coda nel connettore sorgente e di non inviare alcuna risposta a System Plus Evolution. Per far ciò è stata impostata ad *OFF* la Source Queue a *none* da dove prelevare la risposta.

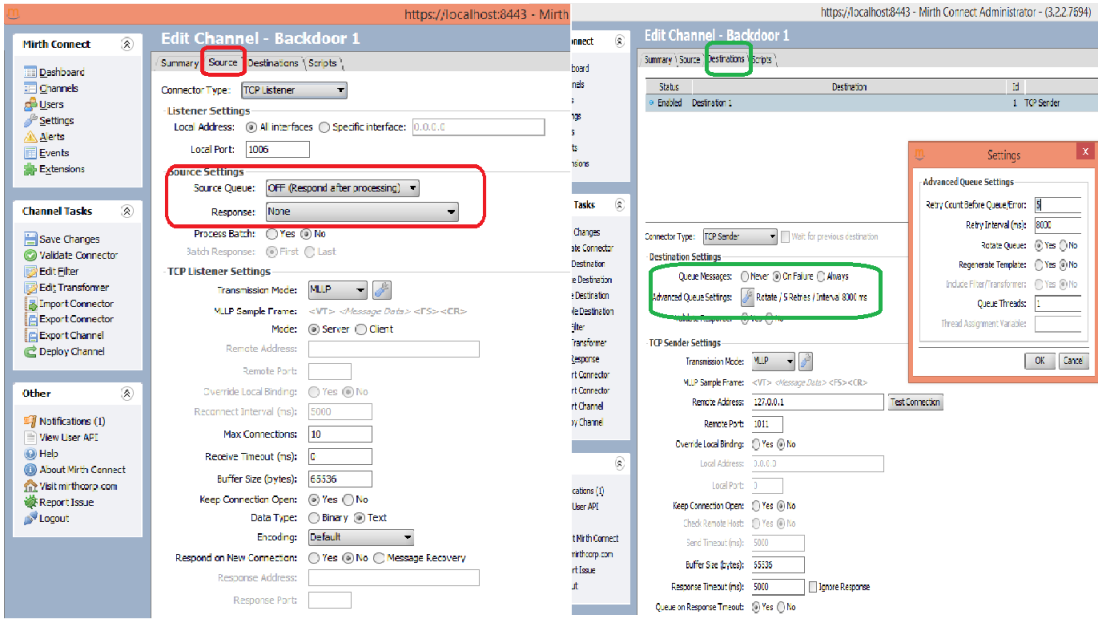

Figura 4.3: boh2

Per le code nel connettore destinazione sono state fatte diverse prove così da capirne appieno il funzionamento, ed infine si è deciso di mettere in coda i messaggi solo se viene fallita la consegna, *On Failure* e se non si riceve alcuna risposta dal sistema in destinazione, *Queue on Response Timeout*. È stato inoltre deciso di provare a reinviare il messaggio altre 5 volte ogni 8 secondi prima di mettere in coda il messaggio, in modo da evitare di metterlo in coda se il sistema in destinazione era solo momentaneamente non in ascolto.

Tali prove sono state essenziali per poter risolvere la problematica della gestione referti nei canali che sono stati sviluppati per l'interfaccia del Sant'Orsola e del San Raffaele.

3. Per quanto riguarda la generazione di un alert è stato sviluppato un codice da inserire nel postprocessor in modo che venisse inviata una mail di warning nel caso in cui si fossero verificati specifiche problematiche quali la ricezione di una risposta negativa. Il codice utilizzato per la generazione di un alert verrà presentato assieme ai canali sviluppati per le interfacce del Sant'Orsola.

4.1.6 Stress Test

Per simulare una grossa mole di messaggi in uscita dal CUP è stato utilizzato il *preprocessor script* in modo da fare un routing interno di un messaggio che viene di volta in volta modifcato. Di seguito si può vedere il codice utilizzato. Anche se i messaggi generati non andavano a buon fine perchè contenevano degli errori sintattici, lo scopo del test era vedere se i canali Mirth erano in grado di far fronte ad un elevata mole di dati.

```
1  var myNewMsg= new XML(SerializerFactory.getSerializer('HL7V2').toXML(message));
2
3  var count=parseInt(myNewMsg['MSH']['MSH.9']['MSH.9.3'].toString());
4  var counter=parseInt(myNewMsg['MSH']['MSH.10']['MSH.10.1'].toString());
5
6
7
8  switch (count){
9     case 0:
10       myNewMsg['MSH']['MSH.9']['MSH.9.1']='ADT';
11       myNewMsg['MSH']['MSH.9']['MSH.9.2']='A01';
12       myNewMsg['PID']['PID.5']['PID.5.1']=[myNewMsg['PID']['PID.5']['PID.5.1'].
             toString()+'lo'];
13       myNewMsg['PID']['PID.5']['PID.5.2']=[myNewMsg['PID']['PID.5']['PID.5.2'].
             toString()+'is'];
14       myNewMsg['MSH']['MSH.9']['MSH.9.3']=count+1;
15       myNewMsg['MSH']['MSH.10']['MSH.10.1']=counter+1;
16       createSegment('EVN', myNewMsg)
17       myNewMsg['EVN']['EVN.1']['EVN.1.1']='A01';
18       delete myNewMsg['OBR'];
19       delete myNewMsg['ORC'];
20       var mess=new XML(SerializerFactory.getSerializer('HL7V2').fromXML(
             myNewMsg));
21       router.routeMessage('HIS', mess);
22       logger.info('case 0')
23       break;
24    case 1:
25       myNewMsg['PID']['PID.5']['PID.5.1']=[myNewMsg['PID']['PID.5']['PID.5.1'].
             toString()+'lo'];
26       myNewMsg['PID']['PID.5']['PID.5.2']=[myNewMsg['PID']['PID.5']['PID.5.2'].
             toString()+'is'];
27       myNewMsg['MSH']['MSH.9']['MSH.9.1']='ORM';
28       myNewMsg['MSH']['MSH.9']['MSH.9.2']='O01';
```

```
29        myNewMsg['MSH']['MSH.9']['MSH.9.3']=count+1;
30        myNewMsg['MSH']['MSH.10']['MSH.10.1']=counter+1;
31        delete myNewMsg['EVN'];
32        createSegment('ORC', myNewMsg)
33        createSegment('OBR', myNewMsg)
34        myNewMsg['ORC']['ORC.1']['ORC.1.1']='NW';
35        var mess=new XML(SerializerFactory.getSerializer('HL7V2').fromXML(
              myNewMsg));
36        router.routeMessage('HIS', mess);
37        logger.info('case 1')
38        break;
39    case 2:
40        myNewMsg['MSH']['MSH.9']['MSH.9.1']='MDM';
41        myNewMsg['MSH']['MSH.9']['MSH.9.2']='T02';
42        myNewMsg['PID']['PID.5']['PID.5.1']=[myNewMsg['PID']['PID.5']['PID.5.1'].
              toString()+'lo'];
43        myNewMsg['PID']['PID.5']['PID.5.2']=[myNewMsg['PID']['PID.5']['PID.5.2'].
              toString()+'is'];
44        myNewMsg['MSH']['MSH.9']['MSH.9.3']=count+1;
45        myNewMsg['MSH']['MSH.10']['MSH.10.1']=counter+1;
46        var mess=new XML(SerializerFactory.getSerializer('HL7V2').fromXML(
              myNewMsg));
47        router.routeMessage('HIS', mess);
48        logger.info('case 2')
49        break;
50    case 3:
51        myNewMsg['MSH']['MSH.9']['MSH.9.1']='ORM';
52        myNewMsg['MSH']['MSH.9']['MSH.9.2']='O013';
53        myNewMsg['PID']['PID.5']['PID.5.1']=[myNewMsg['PID']['PID.5']['PID.5.1'].
              toString()+'lo'];
54        myNewMsg['PID']['PID.5']['PID.5.2']=[myNewMsg['PID']['PID.5']['PID.5.2'].
              toString()+'is'];
55        myNewMsg['MSH']['MSH.9']['MSH.9.3']=0;
56        myNewMsg['MSH']['MSH.10']['MSH.10.1']=counter+1;
57        var mess=new XML(SerializerFactory.getSerializer('HL7V2').fromXML(
              myNewMsg));
58        router.routeMessage('HIS', mess);
59        logger.info('case 3')
60        break;
61
62 }//end switch
```

Poichè il messaggio gestito dallo script è in formato HL7 si è utilizzata la function:

```
1 SerializerFactory.getSerializer('HL7V2').toXML(message);
```

per trasformare il fromato del messaggio da HL7 a XML.
Al messaggio in formato XML sono state apportate le opportune modifiche ,quali il codice di controllo per il messaggio; poi per riportare il messaggio in formato HL7 è stata usata la function:

```
1 SerializerFactory.getSerializer('HL7V2').fromXML(myNewMsg);
```

Infine, per fare il routing interno del messaggio si è utilizzata la function:

```
1 router.routeMessage('HIS', mess);
```

In pratica, quando il messaggio arriva all'HIS questo viene sdoppiato e convertito in formato XML. Sul messaggio sdoppiato vengono fatte le opportune trasformazioni poi viene portato in formato HL7 e fatto il routing interno. Il messaggio originale invece non subisce alcuna alterazione e prosegue il suo corso nel canale.
In questo modo quando arriva un messaggio nel canale questo viene sdoppiato, modificato e reiniviato nel canale creando così un flusso continuo di messaggi sempre diversi. Abbiamo così potuto testare come reagisce il canale di collegamento fra HIS e System Plus ed i risultati sono stati positivi.

4.2 Implementazione interfaccia presso l'ospedale Sant'Orsola di Bologna

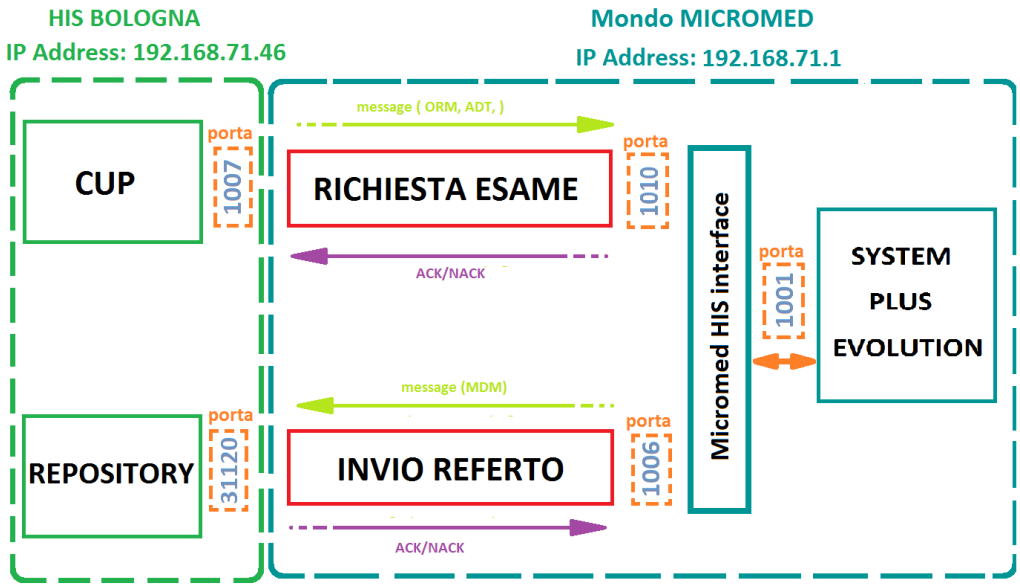

Figura 4.4: Interfaccia implementata, mediante Mirth Connect, nell'Ospedale Sant'Orsola di Bologna

Dopo aver preso confidenza con Mirth Connect si è deciso di provare ad implementare un'interfaccia in una situazione realistica. L'ospedale che è stato scelto è il Sant'Orsola di Bologna poichè la configurazione del SIO è più chiara e semplice rispetto ad altri ospedali, in particolare:

- Il SIO invia i messaggi alla Micromed HIS Interface attraverso un'unica connessione
- Il traffico dei messaggi in entrata alla Micromed HIS Interface è ridotto, rispetto a quello per esempio del San Raffaele di Milano

Per questi motivi si è scelto il Sant'Orsola come ospedale nel quale provare la prima installazione del Mirth, in figura 4.4 si può vedere uno schema degli attori coinvolti nell'interfaccia assieme alle porte e gli indirizzi IP attraverso i quali comunicano.
Per questa prima implementazione dell'interfaccia si è deciso di procedere a passi:

1. Il primo passo è stato quello di interporre fra il SIO e la Micromed HIS Interface 2 istanze del Mirth, un canale RICHIESTA ESAMI e l'altro l'INVIO

REFERTI (vedi figura 4.4), il cui scopo si può intuire dal nome, che facessero semplicemente da postini, senza operare alcuna trasformazione sui messaggi, che però scrivessero un file log nel quale venissero riportati i messaggi che passavano per il canale.

Per adempiere sia alla corretta consegna del messaggio che alla scrittura del file di log i canali sono stati costruiti in questo modo: un connettore sorgente di tipo TCP Listener per ricevere i messaggi e due connettori destinazione, uno TCP Sender per l'invio dei messaggi tramite protocolli TCP/IP ed uno File Writer per scrivere il log file. In figura 4.5 è rappresentato uno stralcio di come si presenta il file log.

```
[2015-8-3 17.22.10]:

MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20140114151219||MDM^T02^MDM_T02|000000004|P|2.5
||||||8859/1|
EVN|T02|20110118101252
PID|1||259856^^^PK||PROVA^AGNESE^^^^^L||19710101|F|||
PV1|1|O|||||||||||||||||||||||||||||||||||||||||||||||||A
ORC|RE|000000101||0000001|CM
OBR|1|||0001^EEG|||||||||||||||||||F
ORC|RE|000000102||0000001|CM
OBR|2|||0002^Descrizione|||||||||||||||||||F
TXA|1|HP|TEXT|||20140318101252|||SPPXRV80A01H501S^SUPPORTO^EXPRIVIA^^^^^^CF|||00030000450003000014|||
||LA||AV|||
OBX|1|ED|00030000450003000014||^application^TEXT^Base64^encoded data|||||F|

[2015-8-3 17.22.10]: SENT : Message successfully sent.

MSH|^~\&|SYSTEMPLUS|MICROMED|OF_EXPRIVIA|EXPRIVIA|20150803172210.710||ACK^T02^ACK|20150803172210.710
|P|2.5 |
MSA|AA|000000004


[2015-8-4 11.11.14]:

MSH|^~\&|OF_EXPRIVIA|EXPRIVIA|SYSTEMPLUS|MICROMED|20140114151219||ORM^O01|000000004|P|2.5|||||8859
/1|
PID|1||259800^^^PK||PROVAprova^BOLOGNA^^^^^L||19710101|F|||VIA RESIDENZA
10^^MILANO^MI^20100^^L^^015146~VIA DOMICILIO
```

Figura 4.5: Stralcio del log file per il canale INVIO REFERTI

Per quanto riguarda la scrittura del file di log in entrambi i canali sono state fatte delle modifiche in modo che venissero riportate sia l'ora esatta in cui il canale riceve il messaggio, sia l'ora la data e lo stato della risposta che si riceve dal sistema in destinazione. Per ricavare queste informazioni sono state inserite le seguenti righe di codice in alcuni punti strategici del canale:

- Nel preprocessor viene definita una variabile per poter andare a capo nelle scrittura del file log e viene salvata una mappa in modo che sia disponibile lungo tutto in ogni momento del canale. Viene inoltre definita e salvata nella mappa una variabile che identifica l'ora in cui il messaggio viene ricevuto dal connettore destinazione, poichè il preprocessor viene eseguito per ogni messaggio arrivato al connettore prima che questo subisca la serializzazione in XML.

```
1 var crRetLnFeed = "\r\n";
2 channelMap.put('CRLF', crRetLnFeed);
3
4 var dateString = DateUtil.getCurrentDate('yyyy-M-d H.m.s');
```

```
5 channelMap.put('Datarrival',dateString);
```

- Per il connettore destinazione File Writer sono stati settati i parametri come in figura 4.6. In questo modo ad ogni messaggio ciò che viene scritto nel file è definito nel *template* e viene aggiunto a ciò che già era contenuto nel file.

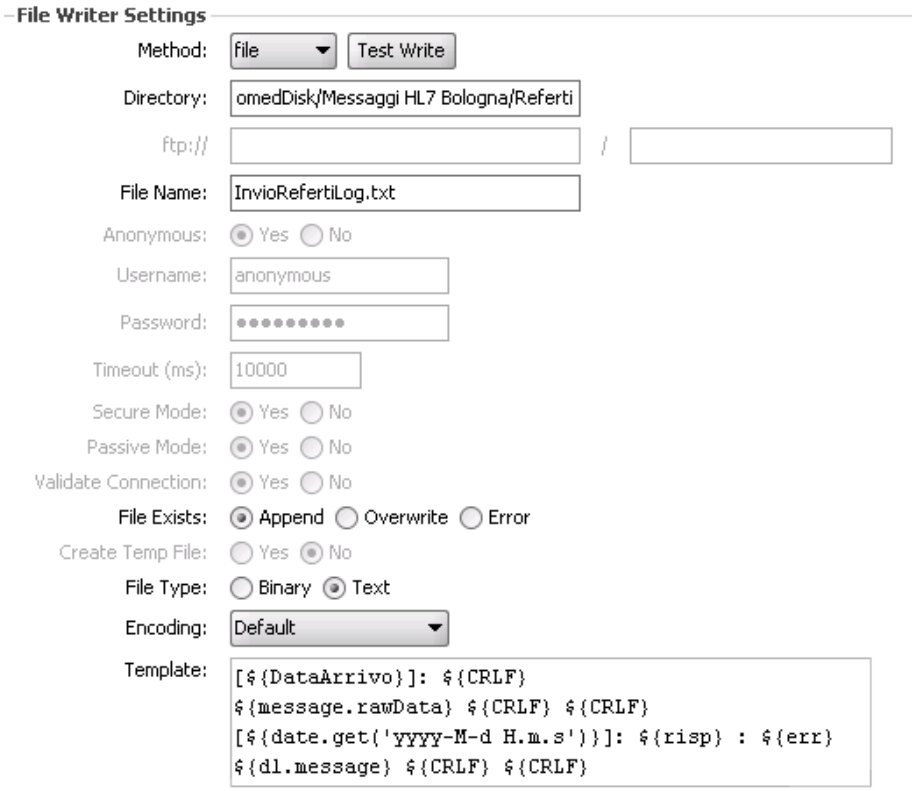

Figura 4.6: Parametri settati nel connettore destinazione File Writer per il canale

- Nel postprocessor script sono state inserite alcune righe di codice per fare un controllo sulla dimensione del file log in modo che non superi i 10 Mega. Di seguito viene riportato il codice.

```
1 var path ='C:/Program Files (x86)/Micromed/MicromedHisInterface/
      InvioRefertiLog.txt';
2
3 var doc=FileUtil.readBytes(path);
4 var size=Math.round(doc.length/1000);
5
6 if (size>10000){
7           FileUtil.write('C:/Program Files (x86)/Micromed/
                MicromedHisInterface/InvioRefertiLog1.txt', false, doc);
8           var del=new java.io.File(path);
9           del["delete"]();
10 }
```

2. Una volta inseriti i canali e visto che il flusso informativo non veniva alterato e che il file di log veniva prodotto correttamente si è passati alla seconda fase in cui sono state fatte delle modifiche al canale INVIO REFERTI per cercare di risolvere il problema per cui principalmente si è deciso di adottare l'utilizzo del

Mirth a Bologna. Tale problema, come già discusso, riguarda l'invio del referto da parte della Micromed HIS Interface. In pratica, accadeva che a fronte di un NACK ricevuto la Micromed HIS Interface non si preoccupasse di dover reinviare il referto, che quindi non veniva depositato nel database ospedaliero.

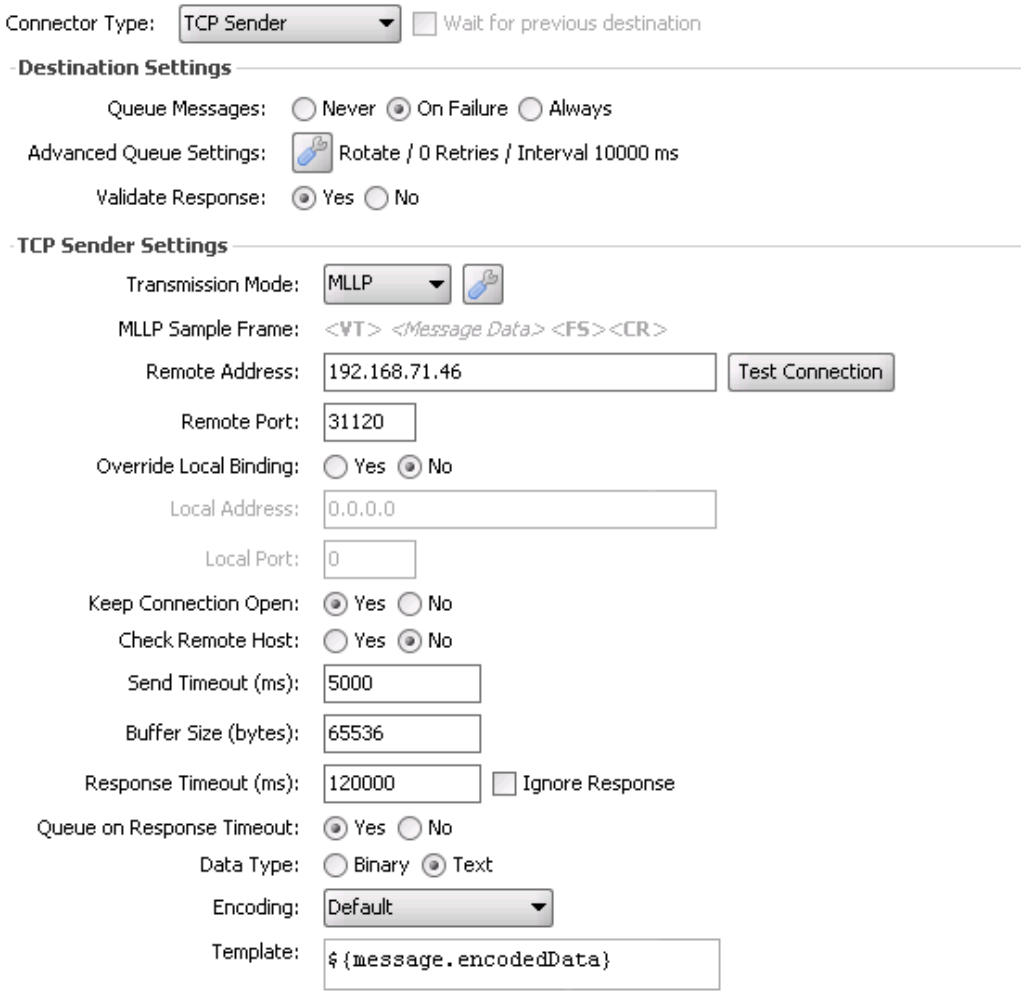

Figura 4.7: Parametri settati nel connettore TCP Sender del canale INVIO REFERTI

Per risolvere questo problema è stato inserito un meccanismo di code all'interno del connettore destinazione TCP Sender in modo che se per un qualche motivo il referto veniva rifiutato dal SIO Mirth si prende in carico il messaggio mettendolo in coda così da poterlo reinviare più tardi. Come si può vedere in figura 4.7, la coda è stata abilitata nel caso in cui fallisca l'invio del referto, questa situazione si può verificare per due motivi: o perchè la connessione è caduta o perchè non viene ricevuta alcuna risposta dal SIO. Per mettere in coda i messaggi nel caso di ricezione di un NACK sono state inserite le seguenti righe di codice nel Response Transformer del connettore.

```
1 if (msg['MSA']['MSA.1']['MSA.1.1'].toString()=='AR'){
2                     responseStatus = QUEUED;
3 }
```

Tale script valuta il campo del segmento MSA in cui è contenuto il codice che valorizza il tipo di Acknowledgment ricevuto (AA, AE, AR) se il codice è AR, poichè le specifiche di Bologna prevedono che la risposta ad un messaggio che

viene rifiutato sia costruita in tal senso, il messaggio viene messo in coda per un successivo tentativo d'invio.

3. Una volta appurato che il canale Mirth inserito avesse risolto il problema, si è deciso per sicurezza di inserire delle righe di codice nel postprocessor per attuare un controllo sul numero di messaggi in errore ed in coda. Se il numero dei messaggi in errore o in coda supera una certa soglia viene generato un alert che invia una e-mail allo staff di Micromed S.p.A. per avvisarli che il numero di messaggi in errore o in coda è cresciuto. In questo modo l'azienda può essere sicura che se sorgono dei problemi nell'interfaccia implementata con il SIO del Sant'Orsola verrà immediatamente avvisata.

```
1  var lastsent = globalMap.get(channelId + '_lastalert');
2  var rightnow = DateUtil.getCurrentDate('dd');
3  var datediff = parseInt(rightnow) - parseInt(lastsent) ;
4
5  //contatori
6  var ercount= ChannelUtil.getErrorCount('44207b51-cca0-4efa-b4dd-1
       b65e5243054',1)
7  var quecount= ChannelUtil.getQueuedCount('44207b51-cca0-4efa-b4dd-1
       b65e5243054',1)
8
9
10
11 if(ercount>=30 || quecount>=25){
12         if (datediff==0) {
13
14                 return;
15
16         }//end if
17         else {
18
19                 //Connessione SMTP ed invio mail
20                 var smtpConn = SMTPConnectionFactory.
                       createSMTPConnection();
21                 smtpConn.setPort(25)
22                 smtpConn.setHost('persdb.aosp.bo.it')
23                 body='Il connettore Destinazione 1 del canale InvioRefertoLog ha
                       '+ercount+' messaggi in errore e '+quecount+' messaggi in
                       coda.\n\r Per favore controllare gli errori';
24                 smtpConn.send('alessandra.casagrande@micromed.eu,nicola.
                       rizzo@micromed.eu,isabella.casagrande@micromed.eu',null, '
                       bologna@micromed.eu', 'Alert: Errored/Queued Counter
                       exceeded', body);
25
26                 globalMap.put(channelId + '_lastalert',rightnow);
27
28         }//end else
29 }//end if
```

Il canale per le richieste esami non è stato modificato in quanto non sono sorte problematiche col suo inserimento nell'interfaccia.

4.3 Implementazione interfaccia presso l'ospedale San Raffaele di Milano

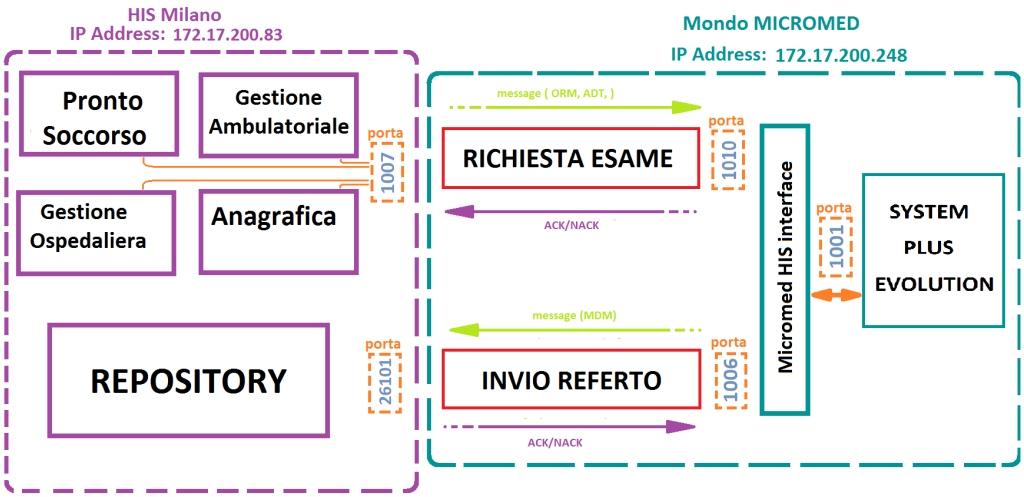

Figura 4.8: Interfaccia implementata, mediante Mirth Connect, nell'Ospedale San Raffaele di Milano

Forti dell'esperienza acquisita al Sant'Orsola di Bologna, si è deciso di utilizzare Mirth Connect in una realtà un po' più complessa, quella del San Raffaele di Milano. La complessità di tale SIO è dovuta al grosso traffico che la Micromed HIS Interface ha in entrata; ciò è dovuto al fatto che la Micromed HIS Interface è connessa contemporaneamente con 4 sistemi che le inviano richieste:

- Il Pronto Soccorso che invia messaggi di tipo $ORM\hat{}O01$
- Gestione Ospedaliera che invia messaggi di tipo $OMG\hat{}O19$, i messaggi di tipo OMG sono utilizzati in alternativa agli ORM ma ne hanno le stesse caratteristiche.
- Gestione Ambulatoriale che invia messaggi di tipo $OMG\hat{}O19$,
- Anagrafica che invia messaggi di tipo $ADT\hat{}A31$ e $ADT\hat{}A40$

Di tutti i messaggi che arrivano alla Micromed HIS Interface, quelli di vero interesse sono gli ORM e gli OMG, perchè relativi a richieste esame da fare con System Plus Evloution, e gli ADT per aggiornare i dati dei pazienti che sono inseriti nell'Evolution Database. Il traffico di messaggi è molto alto perchè come politica aziendale il SIO del San Raffaele è configurato per inviare alla Micromed HIS Interface i messaggi di aggiornamento di tutti i pazienti dell'ospedale, anche di quelli che non sono stati inseriti nell'Evolution Database, per i quali, dunque, non era stato previsto un esame da eseguire con System Plus Evolution.
Per questi motivi il traffico verso la Micromed HIS interface è molto elevato e causava, a volte, il blocco della Micromed HIS Interface. In particolare, accadeva che la Micromed HIS Interface, a volte, richiedesse troppo tempo per elaborare le istruzioni di scrittura del database generate da un messaggio, e di conseguenza non faceva a tempo ad inviare un Acknowledgment al sistema sorgente. Poichè, quest'ultimo, non riceveva alcun Acknowledgment, reinviava il messaggio, questa volta su una nuova connessione, senza però chiudere la precedente. Ma neanche su questa connessione

il sistema sorgente riceveva un Acknowledgment poichè la Micromed HIS Interface era ancora occupata ad eseguire le istruzioni del messaggio inviato la prima volta. Si innescava così una sovrappopolamento di connessioni che mandava in blocco la Micromed HIS Interface. Il SIO, a fronte del blocco della Micromed HIS Interface, vedendosi aumentare i messaggi in coda, poichè, non essendo andati a buon fine, i messaggi venivano spostati in coda per essere reinviati successivamente, avvisava l'azienda che qualche problema era sorto nell'interfaccia, di conseguenza Micromed procedeva al riavvio della Micromed HIS Interface. Una volta riavviata la Micromed HIS Interface, le code del SIO venivano smaltite facendo tornare tutto al normale funzionamento.
Proprio per ovviare al problema del dover riavviare manualmente la Micromed HIS Interface, si è pensato di utilizzare Mirth Connect per gestire le molteplici connessioni in ingresso e convogliarle in unica connessione fra Mirth Connect e Micromed HIS Interface.
Anche in questa realtà si è deciso di inserire due canali del Mirth, una per la RICHIESTA ESAMI ed una per l'INVIO REFERTI. A differenza di Bologna, per l'implementazione di questa interfaccia si è deciso di sviluppare dei canali che avessero solo un connettore destinazione, il TCP Sender, e affidare la scrittura del file log agli script disponibili lungo il canale, in particolare:

- Nel **Preprocessor** è stato inserito il codice, riportato qui sotto, che definisce alcune variabili salvandole nella *Channel Map*, quali ad esempio il percorso della cartella in cui scrivere il file, il controllo che la dimensione del file non superi i 10 Mega e le istruzioni per la scrittura nel file log della data e ora in cui è arrivato il messaggio ed il testo del messaggio.

```
1  //Definizione Variabili
2  var pathlog ='C:/Program Files (x86)/Micromed/MicromedHisInterface/
       RichiestaEsameLog.txt';
3  channelMap.put('PathFileLog',pathlog)
4
5  var pathOldlog ='C:/Program Files (x86)/Micromed/MicromedHisInterface/
       RichiestaEsameLogOLD.txt';
6  channelMap.put('PathFileLogOLD',pathOldlog)
7
8  var crRetLnFeed = "\r\n";
9  channelMap.put('CRLF', crRetLnFeed);
10
11 //--------------------------------------
12 //Controllo che il file di log non superi i 10 mega
13 FileUtil.write(pathlog, true,crRetLnFeed);
14
15 var doc=FileUtil.readBytes($('PathFileLog'));
16 var size=Math.round(doc.length/1000);
17
18 if (size>10000){
19
20          FileUtil.write($('PathFileLogOLD'), false, doc);
21          var del=new java.io.File($('PathFileLog'));
22          del["delete"]();
23 }
24
```

```
//--------------------------------------
//Scrittura del log file
var datetimeArrivoMSG = DateUtil.getCurrentDate('yyyy-M-d H.m.s');

FileUtil.write(pathlog, true,datetimeArrivoMSG+crRetLnFeed);
FileUtil.write(pathlog, true,message.toString()+crRetLnFeed);
```

- Nel **Postprocessor** è stato inserito il codice per la scrittura nel file log della data e ora in cui è stata ricevuta la risposta dal sistema in destinazione e il testo della risposta ricevuta.

```
//Prelievo data, ora e testo della riposta ricevuta dal sistema in destinazione
var datetimeACKNACK = DateUtil.getCurrentDate('yyyy-M-d H.m.s');

var rispget=responseMap.get('Destination 1').getStatus()
var errget=responseMap.get('Destination 1').getStatusMessage()

var destination = responseMap.get('Destination 1');
var responseMessage = destination.getMessage();

FileUtil.write($('PathFileLog'),true,datetimeACKNACK+' '+rispget+': '+errget+
    $('CRLF'));
FileUtil.write($('PathFileLog'),true,responseMessage + $('CRLF')+$('CRLF'));
```

Poichè al San Raffele di Milano i problemi risultavano soprattutto in ingresso alla Micromed HIS Interface, di seguito vediamo le impostazioni che sono state scelte per il canale RICHIESTA ESAME. Come si può vedere in figura 4.9, sono stati settati i seguenti parametri per il connettore sorgente:

- Prelevare la risposta che è stata definita nel postprocessor per inviarla al sistema sorgente. Tale scelta è dovuta al fatto che in alcuni casi inviare la risposta dalla Destinazione 1, come a Bologna, può generare dei problemi. In particolare nel caso in cui la Micromed HIS Interface ci metta troppo ad elaborare le istruzioni di un messaggio, il canale Mirth non riceve nessuna risposta da poter inoltrare al SIO, ma autogenera una risposta vuota con la quale classifica il messaggio, all'interno del Mirth, come ERRORED. Tale risposta vuota è proprio quella che viene inviata al sistema sorgente se non viene ricevuta, per un qualche motivo, una risposta dal sistema in destinazione. La differenza, rispetto al Sant'Orsola, per cui si è scelto di prelevare la risposta dal postprocessor e non dalla destinazione 1, è che il SIO del San Raffaele, a fronte di una risposta vuota, reinvia il messaggio un certo numero di volte, dopodichè lo considera come inviato correttamente, in quanto una risposta, seppur vuota, è stata ricevuta. Ciò però non è rappresentativo di quello che è successo realmente, infatti non è noto se la Micromed HIS Interface abbia completato con successo l'esecuzione del messaggio. Per ovviare a questo problema si è dunque deciso di inviare la risposta che viene definita nel postprocessor in modo che se il canale Mirth non riceve alcuna risposta allora neanche il SIO deve riceverla, se invece viene ricevuta la si inoltra. Per simulare questo comportamento è stato inserito il seguente codice all'interno del postprocessor, che controlla se la risposta salvata nella destinazione 1 è vuota. In base al risultato di questa interrogazione viene scelto che risposta inviare:

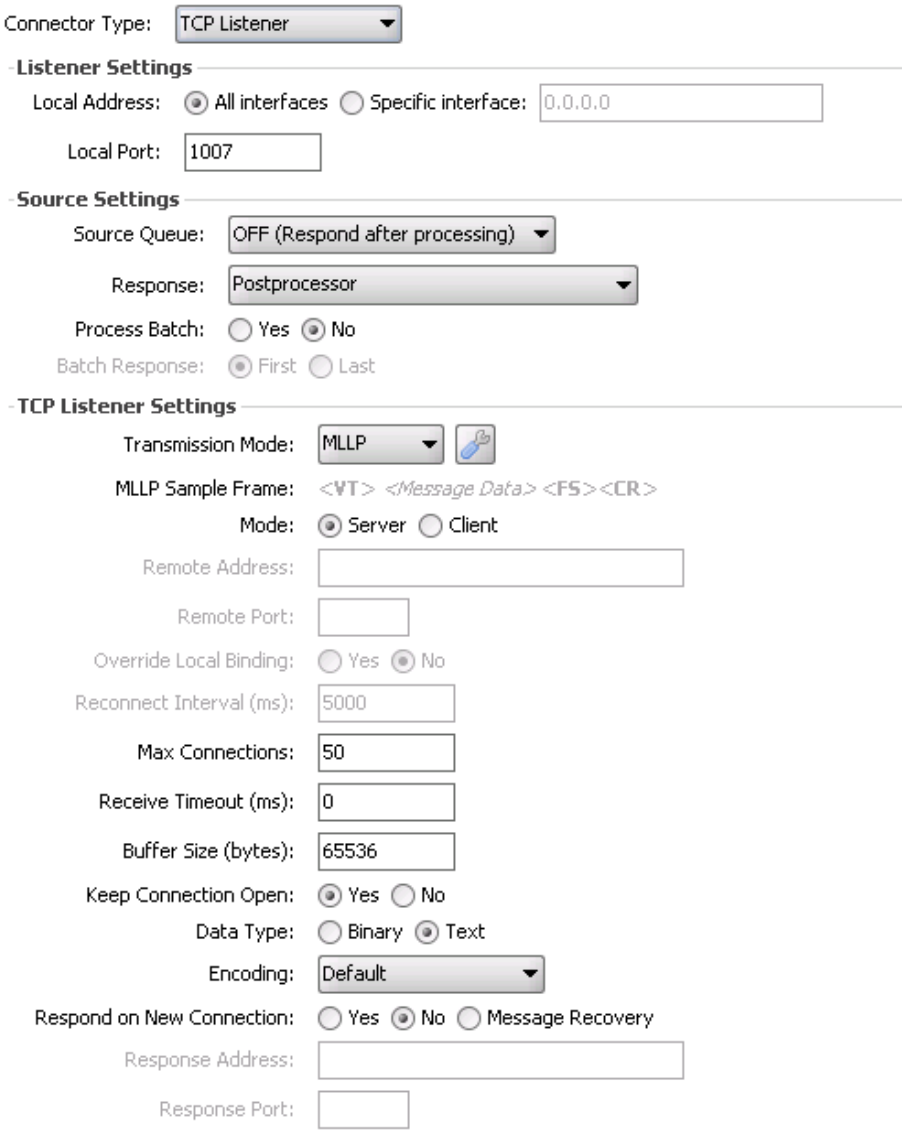

Figura 4.9: Parametri del connettore TCP Listener del canale Richiesta Esame del San Raffaele

```
1 if($r('Destination 1').getMessage().length()==0){
2     return;
3 }
4 else{
5     return $r('Destination 1');
6 }
```

- Il massimo numero di connessioni in ingresso è stato aumentato a 50, in modo da poter avere un alto margine sul numero di connessioni che si possono instaurare fra SIO e Mirth senza che insorgano problemi

- Infine si è abilitato il keep connection open in ingresso, in modo tale che le connessioni stabilite dal SIO rimangano aperte finchè il SIO stesso non le chiude. Tale scelta è stata fatta in base al fatto che il SIO del San Raffaele, una volta stabilita la connessione, la mantiene aperta per poter inviare più messaggi. Tale scelta aumenta le performance del canale poichè, ogni volta che viene stabilita una connessione con protocollo TCP fra due dispositivi,

vengono spesi alcuni millisecondi per assicurare una connessione vicendevole e sicura. Per la comunicazione fra due dispositivi che hanno un traffico elevato, come nel caso del San Raffaele, mantenere la connessione aperta una volta che è stata stabilita, è una scelta che permette ai due dispositivi di non dover perdere tempo per stabilire ogni volta una connessione sicura.

Per la connessione fra Mirth e la Micormed HIS Interface sono stati settati i seguenti parametri nel connettore destinazione, come si può vedere in figura 4.10:

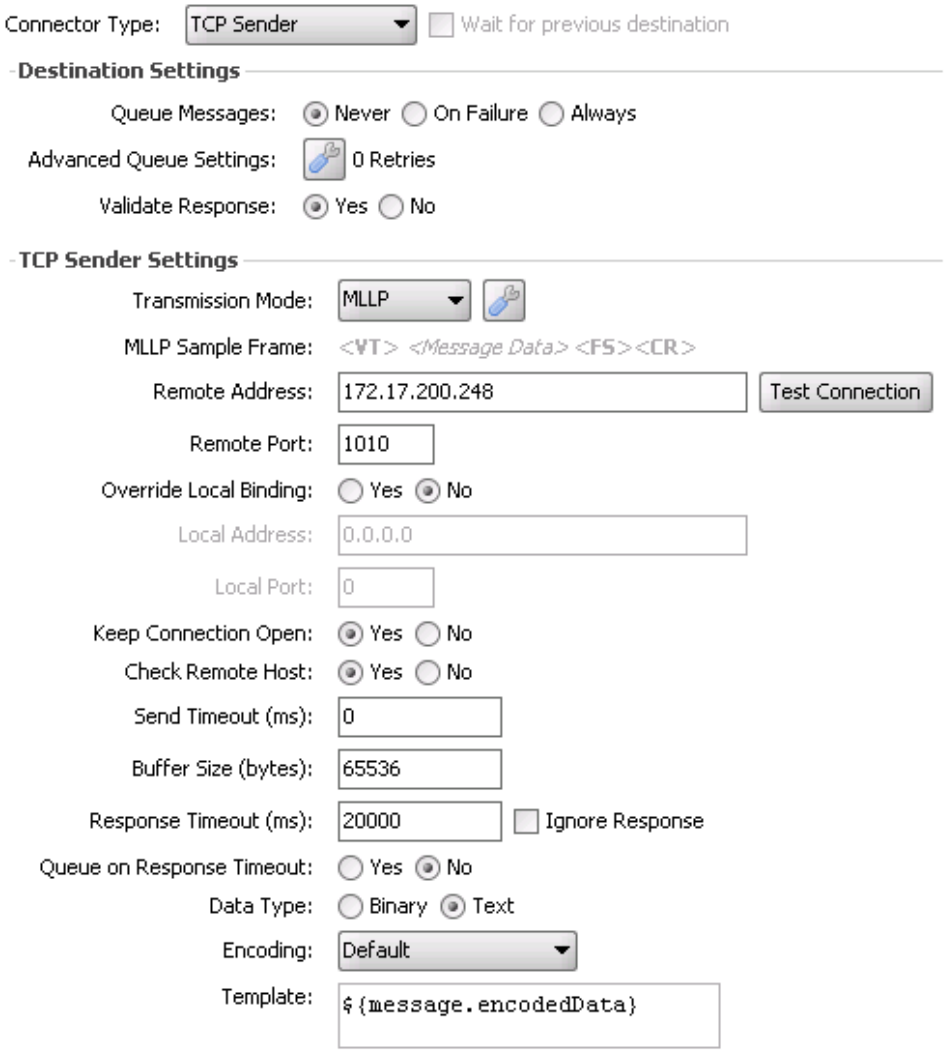

Figura 4.10: Parametri del connettore TCP Sender del canale Richiesta Esame del San Raffaele

- Non sono state abilitate le code poichè è compito del SIO reinviare un messaggio a fronte di un NACK o di una mancata risposta del sistema in destinazione
- Per quanto riguarda il mantenere la connessione aperta o chiuderla dopo l'invio di ogni messaggio si è deciso di adottare la stessa politica del SIO in quanto il traffico in uscita dal canale verso la Micromed HIS Interface è comunque elevato. Per utilizzare questa politica è stato abilitato il Keep Connection Open e posto a 0 il Send Timeout, tempo in cui la connessione deve rimanere aperta per l'invio del messaggio. Ricordiamo che porlo a zero equivale a mantenere aperta per un tempo indefinito la connessione.

- Una problematica che sorge dal mantenere la connessione aperta ed inviarvici più messaggi è che non si può essere certi che il sistema in destinazione sia in ascolto. Per ovviare a questo problema è stata abilitata l'opzione *Check Remote Host* che controlla, prima dell'invio di ogni messaggio, che il sistema in destinazione sia effettivamente in ascolto.

Per quanto riguarda il canale Invio referti è stata adottata una configurazione uguale a quella utilizzata a Bologna eccetto per la messa in coda dei messaggi nel caso in cui vengano rifiutati dal SIO, poichè al San Raffaele non sussiste ancora tale problematica.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Risultati del lavoro svolto

Nel corso di questa tesi sono state esaminate alcune delle problematiche che possono insorgere nell'implementazione di un'interfaccia che possa assicurare l'interoperabilità fra due dispositivi con caratteristiche hardware e software diverse. In particolare, si sono analizzate le problematiche che Micromed S.p.A. aveva riscontarto nella realizzazione di interfacce tra il loro applicativo System Plus Evolution ed il SIO di alcuni ospedali e si è poi valutato se Mirth Connect, affiancato alla Micromed HIS Interface, potesse essere uno strumento utile a risolvere tali problematiche garantendo l'interoperabilitá fra il SIO e l'applicativo Micromed.
In particolare, l'azienda aveva riscontrato delle problematiche nell'integrazione in due particolari situazioni Ospedaliere: al Sant'Orsola a Bologna ed al San Raffaele a Milano. Tali problematiche si possono essenzialmente riassumere in due punti:

- La gestione corretta e sicura nell'invio del referto per il Sistema Informativo Ospedaliero di Bologna
- La gestione di molteplici connessioni in ingresso per le richieste esame nel Sistema Informativo Ospedaliero di Milano
- la gestione degli Acknowledgment
- la scrittura di un file log che tenga traccia dei messaggi in arrivo ed in uscita alla Micromed HIS Interface.

La prima fase della tesi, svolta in gran parte in azienda, è stata incentrata sulla valutazione di Mirth Connect. Si è cercato di capire se effettivamente potesse aiutare a risolvere le problematiche sopra elencate, per far ciò è stato dunque necessario prendere confidenza con il programma focalizzandosi sulle sole funzionalità che potessero essere utili alla risoluzione di tali problematiche. In particolare, sono stati esaminati e approfonditi i seguenti argomenti:

- Workflow del messaggio nel canale
- Funzionalità ed utilizzo degli script messi a disposizione nel Workflow del canale: Deploy, Preprocessor, Postprocessor, Undeploy, Edit Filter, Edit Transformer, Edit Response.
- Javascript

- Modalità di utilizzo delle code sia per il connettore Sorgente che per il Destinazione

- Parametri settabili per adattare i connettori TCP Sender, TCP Listener e File Writer alle differenti situazioni che si devono affrontare

Una volta completata la fase di valutazione, presentati i risultati ottenuti e mostrato le potenzialità di Mirth Connect come strumento per garantire l'interoperabilità fra due dispositivi, mi è stato affidato il compito di sviluppare due canali da utilizzare per cercare di risolvere i problemi riscontrati nell'integrazione con il SIO di Bologna. I canali sono stati sviluppati in ambiente di test e successivamente implementati nella realtà ospedaliera con le modalità già presentate nella sezione 5.2.1. In particolare, si è deciso di reinviare il referto se si verifica uno dei seguenti eventi:

- il canale Mirth non riesce a connettersi al Repository del SIO

- il Repository del SIO non invia alcuna risposta di avvenuta ricezione del messaggio

- il Repository invia un NACK in cui il segmento MSA è valorizzato dal codice AR

Questa soluzione, una volta validata e messa in produzione ha garantito l'interoperabiltà fra il SIO del Sant'Orsola e System Plus Evolution.
La seconda implementazione riguarda le problematiche riscontrate nell'integrazione con il SIO del San Raffaele di Milano. Per questa implementazione mi è stato chiesto di valutare un'interfaccia, sviluppata tramite Mirth Connect, che potesse principalmente risolvere i problemi generati dalle molteplici connessioni in ingresso. Anche in questo caso sono stati valutati prima i canali in un ambiente di test e successivamente si è passati in produzione. Inizialmente, nella fase di test le problematiche per cui erano stati sviluppati i canali hanno continuato a verificarsi, ma grazie al log file tracciato dai canali Mirth Connect, è risultato chiaro che nei casi in cui la Micromed HIS Interface, per un qualche motivo, mancava di inviare l'acknowledgment, Mirth autogenerava ed inviava una risposta vuota all'HIS e quest'ultimo considerava tale risposta come valida. Tale problema è stato risolto, come già presentato nella sezione 5.2.2, selezionando nel postprocessor del canale Mirth la risposta da inviare all'HIS in base al contenuto della risposta ricevuta dalla Micromed HIS Interface. Ad oggi i canali implementati nell'integrazione con il SIO del San Raffaele risolvono egregiamente il problema per cui sono stati sviluppati.
In conclusione, posso affermare che Mirth Connect ha risolto le problematiche d'interoperabilità riscontrate dall'azienda soddisfando le aspettative che aveva nei sui confronti, e che, con la sua versatilità, è riuscito a garantire l'interoperabilità fra System Plus Evolution ed il SIO, sia del Sant'Orsola che del San Raffaele portando a termine tutti gli obiettivi che all'inzio della tesi erano stati fissati.

5.2 Sviluppi futuri

Nell'interfaccia giá implementata al San Raffaele di Milano non è ancora stato inserito alcun alert che segnali a Micromed il verificarsi di eventuali errori nel canale Mirth. Il motivo per cui non è ancora stato inserito tale alert é dovuto al fatto che

il SIO di Milano non ha ancora fornito a Micromed i dati del server di posta a cui connettersi per poter inviare le e-mail.
Tuttavia grazie ai risultati ottenuti, l'azienda adotterà Mirth Connect per affiancare la Micromed HIS Interface in maniera da garantire l'interoperabilità fra System Plus Evolution e i SIO che richiederanno l'implementazione di un interfaccia.
Dallo studio di Mirth Connect è risultato che un ulteriore sviluppo per Micromed potrebbe essere, in futuro, l'utilizzo del solo Mirth Connect per assicurare l'interoperabilità fra System Plus Evolution ed il SIO, abbandonando, quindi, l'uso della Micromed HIS Interface e trasferendo i compiti di cui si occupava, quali ad esempio la scrittura nel database, a Mirth Connect. Tale soluzione é possibile in quanto la versatilità di Mirth Connect permette la scittura nei database attraverso il connettore *Database Writer*.

Bibliografia e Siti Web Consultati

Bibliografia

[1] Bevenuto Nevio, Zorzi Michele *Principles of Communications Networks and Systems*, Ed. Wiley, 2011

[2] Bracchi Giampio, Francalanci Chiara,Motta Gianmario, *Sistemi informativi d'impresa*, Ed. Mc Graw-Hill, 2010.

[3] Buccoliero Luca, Mattavelli Elisa, *ECDL Health*, Ed. Tecniche Nuove, 2008.

[4] Henderson Mike, *HL7 Messaging : Second Edition*, Tech, 2007

[5] Kuhn K. A., Giuse D. A., *From hospital information systems to health information systems - problems, challenges, prospectives*, Yearbook of Medical Informatics, pp. 63-73, 2001

[6] Nikolassy Riccardo, *Programmare in Javascript*, Ed. Hoepli, 2007

[7] Pinciroli Francesco, Combi Carlo, Pozzi Giuseppe, *Basi di dati per l'informatica medica. Concetti, linguaggi, applicazioni*, Ed. Pàtron, 1998

[8] Pisanelli Domenico, Sicurello Francesco, Ricci Fabrizio, *Dall'informatica medica alla sanità elettronica: lezioni dal passato e prospettive per il futuro*, Ed. Edisef, 2012

[9] Samsone Mario, *Introduzione ai Sistemi Informativi Sanitari*, Ed. Laterza, 2005

[10] Teti Antonio, Festa Giuseppe, *Sistemi Informativi per la Sanità* ,Ed. Apogeo, 2009.

[11] Winter Alfred, Haux Reinhold, Ammenwerth Elske, Brigl Birgit, *Health Information Systems, Architectures and Strategies*, Ed. Springer, 2011

Siti Web Consultati

[1] *www.corepointhealth.com*

[2] *www.HL7.org*

[3] *www.hl7italia.it*

[4] *www.html.it*

[5] *www.mirth.com*

Ringraziamenti

Giunto a questo traguardo è dovoroso dedicare un pensiero a tutte le persone che mi hanno sostenuto e aiutato in questo percorso di laurea e di vita.

Ringrazio innanzitutto il professor Sparacino, per la professionalità e l'esperienza con cui mi ha guidato, e tutti i dipendenti di Micromed S.p.A. per l'accoglienza ed il supporto datomi in questi ultimi mesi.

Ringrazio i miei amici, i compagni d'Università e tutte le persone che mi hanno sostenuto, sopportato e hanno condiviso con me frustrazioni e risate, fuori e dentro le aule del DEI.
In particolare ringrazio i miei colleghi Rigo, Chiara, Eugi, Anahita, Teo ed Alessia per aver reso il tempo passato a Padova un piacere, spero che con la fine di questo periodo non finisca anche il legame che si è creato nel corso di questi anni.
Un caloroso grazie è rivolto ai miei amici, che standomi accanto in questi anni sono diventati dei fratelli e che continuerò sempre a considerare come tali.

Ringrazio immensamente la mia famiglia per avermi amato, coccolato e fatto crescere. Con tutto il cuore ringrazio i miei 32 nipoti ed i miei fratelli Chiara, Giovanna, Stefano, Giacomo, Giuditta e Mariamaddalena per essere sempre un esempio ed un modello al quale aspirare.
Anche se il ringraziamento piú grande va a voi, Mamma e Papà, per avermi dato la vita, per essermi sempre stati accanto anche quando non me ne accorgevo e per tutti i sacrifici fatti per farmi raggiungere questo traguardo.

Infine un ringraziamento speciale va a te, Elisabetta, per l'amore e la pazienza con cui mi sei stata accanto in questi anni, per essere stata il motore che mi ha spinto a non abbandonare e l'obiettivo a cui puntare. Con la speranza e la gioia di poter raggiungere nuovi traguardi assieme.

Grazie di cuore,
Gregorio Canal