



Sviluppo FrameWorks CPL , integrazione con MJSIP, CpleEd. **JCPL**

Autori

Antonio Di Fluri e Marco Maltraversi

Inizio progetto	02-01-2007
Fine progetto	28-03-2007
Versione software	2.0.1
Versione relazione	2.0.1
Dati aggiuntivi	CplEd

1. Introduzione

CPL è un linguaggio di scripting usato per definire come devono venire gestite le chiamate entranti o uscenti in una rete SIP.

Questo linguaggio è basato su XML ed è stato sviluppato per essere eseguito all'interno di un proxy SIP, o all'interno di un terminale con funzionalità avanzate, per implementare nuovi servizi. CPL è stato concepito per essere utilizzato da utenti considerati anche non fidati, per inserire i loro servizi sui server SIP.

CPL è un linguaggio leggero, efficiente, facile da implementare ed estendibile perché è possibile aggiungere nuove funzionalità personalizzate in modo che, in ogni caso, gli script già esistenti possano continuare a funzionare.

CPL permette di lavorare con una delle principali funzionalità di una rete: il Call Control.

CPL può essere utilizzato per implementare servizi in differenti scenari:

1. si possono utilizzare script creati direttamente dagli utenti in modo che possano personalizzare la politica della gestione delle proprie chiamate inviando questi al server
2. gli script possono venir creati dagli amministratori di sistema secondo le preferenze degli utenti
3. gli script possono venir generati da applicazioni web che traducono le richieste in documenti CPL.

Dato che CPL è un linguaggio standard può essere utilizzato da terze parti per creare servizi personalizzati per i clienti, e questi servizi possono venire eseguiti su server propri dei clienti, oppure direttamente dai fornitori di servizi di questi. CPL è un linguaggio abbastanza maturo ed è definito dall'RFC3880 e RFC2824. Esso comunque non permette di generare delle chiamate verso due o più utenti perché gli script vengono attivati solo ed esclusivamente in seguito ad eventi relativi a chiamate, come la ricezione o l'invio, e non può essere utilizzato per descrivere dei comportamenti complessi, benché il linguaggio in se sia comunque flessibile.

È stato pensato per essere semplice, estendibile, facilmente maneggiabile da programmi attraverso interfacce grafiche, slegato da qualunque piattaforma o dal protocollo di segnalazione utilizzato (che normalmente si tratta di SIP o H.323); Oltre a ciò si è tenuto conto delle problematiche di sicurezza che possono nascere in questi contesti, e dato che gli utenti possono creare liberamente i propri profili CPL sotto forma di script e inviarli ad un server che li esegue, è stato fatto in modo che questi script non possano eseguire programmi arbitrariamente, inoltre, non esiste il concetto di variabile o di ciclo, quindi vengono automaticamente esclusi tutti i problemi legati a situazioni di *overflow* e *denial of service*.

2 La struttura di CPL

La struttura gerarchica di XML viene rispecchiata in CPL: tutto lo script `e racchiuso all'interno del tag radice <cpl>, all'interno di questo vengono inserite le azioni di controllo delle chiamate, che possono essere di tre tipi: <subaction>, <incoming> e <outgoing>, questi vengono definiti tag di azione. Gli elementi di tipo *subaction* definiscono quelle che in un generico linguaggio di programmazione vengono considerate funzioni, cioè frammenti di codice che possono venir chiamati da qualunque altra parte del programma. La chiamata ad una *subaction* in CPL viene fatta attraverso il tag <sub>. A differenza della maggior parte dei linguaggi di programmazione non è possibile richiamare più volte la stessa funzione all'interno di una stessa esecuzione, quindi non è possibile creare programmi ricorsivi. I tag di tipo *incoming* e *outgoing* definiscono le parti di script che vengono chiamate quando arriva in ingresso una chiamata (*incoming*) o quando sta uscendo una chiamata dal sistema *outgoing*. In generale `e normale che sia presente uno solo di questi elementi all'interno di uno stesso script, questo perchè viene utilizzato uno o l'altro a seconda del contesto in cui questo viene eseguito.

2.1 Nodi CPL

2.1.1 Nodi di switch

Vengono utilizzati per eseguire determinate azioni subordinate all'avverarsi di una determinata condizione, quindi a seconda della situazione in cui è stata fatta una chiamata vengono effettuate delle operazioni piuttosto che altre. Ad esempio il seguente frammento di codice:

```
<address-switch field="origin">
<address is="sip:webmaster@google.it">
... ramo 1 ...
</address>
<address subdomain-of="siti.org">
... ramo 2 ...
</address>
<otherwise>
... ramo 3 ...
</otherwise>
</time-switch>
```

Discrimina a seconda dell'indirizzo chiamante il comportamento del proxy, il nodo è di tipo <address-switch> che si occupa di elaborare l'indirizzo di una chiamata, in particolare in questo caso viene utilizzato l'indirizzo di origine (specificato nel suo attributo field="origin"). Una volta specificato su cosa verterà la scelta vengono presentati i vari casi, ognuno dei quali corrisponderà ad un output. Dato che stiamo utilizzando un <address-switch>, gli output saranno di tipo <address>, la condizione effettiva viene specificata negli attributi di ogni output. In particolare nel primo address viene controllato che l'indirizzo corrisponda a sip:webmaster@google.it, nel qual caso verrebbe eseguito il codice presente nel ramo 1; la seconda condizione viene verificata (ovviamente solo nel caso in cui la prima non sia vera) se il chiamante appartiene al dominio siti.org, nel cui caso l'esecuzione passa alle istruzioni presenti nel ramo 2. Esistono degli output che sono presenti all'interno di ogni nodo di switch, uno di questi corrisponde al tag <otherwise> che viene eseguito quando nessuno degli altri output risulta verificato. Nei linguaggi di programmazione classici corrisponde all'elemento else del costrutto if then else. Un altro output particolare è il <not-present> che viene scelto quando il parametro presente nel tag di switch non è presente all'interno della richiesta della chiamata.

Ovviamente non è applicabile per tutti i casi, perché alcune informazioni, come ad esempio l'ora di arrivo della chiamata, sono sempre presenti.

Address Switch

Viene utilizzato per prendere decisioni in funzione degli indirizzi presenti nella richiesta di chiamata.

Il tag specifico di questo nodo è:

Tag: <address-switch>

Output: address, otherwise, not-present

Parametri: field, subfield

Attraverso i parametri field e subfield si può specificare quale indirizzo utilizzare per eseguire i controlli, in particolare nel primo parametro, che è obbligatorio, si possono specificare i valori: origin, destination e original-destination, che corrispondono rispettivamente all'indirizzo di chi richiede la chiamata, l'indirizzo di chi riceve la chiamata e l'indirizzo originale del destinatario presente nella richiesta di INVITE nel caso si siano modificati gli indirizzi di destinazione.

Nel parametro subfield si può decidere se analizzare solo una parte dell'indirizzo, in particolare i valori validi possono essere: address-type, user, host, port, tel, display.

In particolare corrispondono a:

address-type tipo di indirizzo, in particolare può assumere i valori: sip, tel (nel caso sia un numero di telefono vero e proprio) o h323 (se il protocollo di segnalazione non fosse SIP ma H.323).

user nome utente

host nome della macchina presente nell'indirizzo

port numero di porta specificata nell'indirizzo

tel il numero telefonico, nel caso sia un indirizzo di tipo telefonico

display corrisponde alla descrizione testuale che appare nell'indirizzo

Ogni output di questo switch specifica una determinata condizione applicabile all'elemento dell'indirizzo scelto, in particolare ad ogni tag address presente all'interno di questo nodo specifica un condizione ben precisa. In particolare viene utilizzato un attributo per specificare il tipo di condizione. Gli attributi a disposizione utilizzabili all'interno di ogni output address sono:
is viene utilizzato per specificare una corrispondenza esatta del valore specificato all'interno di questo attributo con l'indirizzo selezione nell'address-switch *contains* viene verificato quando il valore specificato è contenuto nell'indirizzo (è possibile utilizzarlo soltanto con il subfield display)
subdomain-of può essere utilizzato per controllare se un determinato indirizzo appartiene ad un determinato dominio

String Switch

Viene utilizzato per fare scelte su informazioni di tipo testuale presenti in vari campi dell'intestazione del pacchetto INVITE, come l'oggetto del messaggio o il tipo di UA utilizzato.

Il tag utilizzato è:

Tag: <string-switch>

Output: string,otherwise,not-present

Parametri: field

Il parametro field contiene una stringa con la descrizione del campo dell'intestazione a utilizzare per i confronti, in particolare è possibile inserire un valore tra:subject, organization, user-agent e display.Nel dettaglio questi campi corrispondono a:

subject definisce il campo dell'oggetto della comunicazione.

organization corrisponde all'organizzazione di chi sta eseguendo la chiamata

user-agent è il nome del UA che viene utilizzato da chi effettua la chiamata

display corrisponde alla descrizione testuale che appare nell'indirizzo 'output principale utilizzato in questo switch è string, che in base ai parametri specificati controlla il contenuto del campo specificato, in particolare può contenere li attributi:

is cerca una corrispondenza esatta del valore specificato in questo attributo con il campo selezionato *contain* controlla se il valore specificato nell'attributo è contenuto nel campo. I tre all'output string è possibile utilizzare, come quanto già visto per lo switch precedente, anche gli output otherwise e not-present con lo stesso significato.

Language Switch

Questo switch esegue i controlli in base alla lingua utilizzata dal chiamante, lo switch utilizzato è:

Tag: <language-switch>

Output: language, otherwise, not-present

In realtà quando si parla di lingua utilizzata, ci si può riferire alla lingua che il chiamante potrebbe preferire per la comunicazione. Questo nodo non necessita di alcun parametro.

L'output utilizzato è language, in cui viene specificato il tipo di lingua in base al contenuto del parametro matches. Il formato utilizzato per specificare la lingua è definito nel RFC 3066, è possibile definire anche dei gruppi di lingue, in modo da far verificare per un solo output diversi linguaggi contemporaneamente.

Oltre all'output language è possibile utilizzare, come già visto, anche gli output otherwise e not-present con i medesimi significati.

Time Switch

Attraverso questo switch è possibile specificare delle condizioni basate sul momento di arrivo di chiamata, potendo definire degli intervalli di tempo in modo estremamente flessibile e potente.

Il tag che definisce questo switch è:

Tag: <time-switch>

Output: time, otherwise

Parametri: tzid, tzurl

Con questo switch è possibile creare degli script che si comportino in modo diverso a seconda dell'ora o del giorno in cui arriva la chiamata. È possibile specificare intervalli di tempo utilizzando lo standard del Internet Calendar and Scheduling Core Object Specification (iCalendar COS) definito all'interno del RFC 2445, in cui vengono definiti due formati, uno per specificare un istante di tempo o un'indicazione temporale (un dato giorno o mese, per esempio), mentre il secondo viene utilizzato per rappresentare delle durate. Utilizzando combinazioni di questi due elementi è possibile definire una serie di intervalli di tempo, anche ricorrenti, in modo estremamente flessibile. All'interno del tag time-switch è possibile definire due attributi: tzid e tzurl, che possono contenere rispettivamente un Time Zone Identifier e un Time Zone URL, definiti entrambi nel RFC 2445, vengono usati per specificare a quale zona oraria appartengono le indicazioni di tempo gestite da questo switch. Non specificando questi attributi si dà per scontato che tutte le ore specificate vengono calcolate in base alle impostazioni locali del sistema nel quale vengono utilizzati i profili.

Gli output di questo switch sono definiti dal tag time, per ognuno di questi output è possibile definire una serie di intervalli di tempo. Il modo più semplice per utilizzare questo tag è specificare un singolo intervallo di tempo, questo può essere fatto in due modi, indicando l'istante di partenza e l'istante finale (usando gli attributi dtstart e dtend), oppure l'inizio e la durata dell'intervallo (con gli attributi dtstart e duration).

Indicazioni più complesse possono essere costruite utilizzando degli intervalli ricorrenti, questo viene fatto in primo luogo specificando l'attributo freq in cui viene specificata la frequenza di ripetizione. I valori utilizzabili possono essere: secondly, minutely, hourly, daily, weekly, monthly o yearly, in questo modo si dichiara che l'intervallo di tempo specificato viene considerato ripetuto rispettivamente ogni secondo, ogni minuto, ogni ora, ogni giorno, ogni settimana ogni mese ed ogni anno.

E' importante fare attenzione a non definire intervalli di durata tale che si sovrappongano durante le ripetizioni, cioè se si vuole che un intervallo venga ripetuto ogni ora, questo non deve avere durata superiore all'ora. E' possibile specificare ogni quanto tempo spesso l'intervallo viene ripetuto attraverso il parametro interval, cioè se vogliamo definire un intervallo che venga ripetuto ogni 2 settimane dovremmo inserire gli attributi:

... freq="weekly" interval="2" ...

mettendo il suo valore pari a 3 questo viene ripetuto ogni 3 settimane. Se viene omissa l'attributo freq, ma vengono impostati degli altri attributi relativi alla ricorrenza degli intervalli, viene data per scontata una frequenza di tipo giornaliero. Se si vuole dare un termine alla ripetizione degli intervalli si può utilizzare il parametro until, oppure count; il primo permette di specificare l'ultimo istante in cui viene considerato valido un intervallo, mentre il secondo permette di definire il numero di volte che questo deve venire ripetuto. Esiste una serie di attributi con cui è possibile specificare delle liste di valori di tempo (ad esempio di giorni o di ore) in cui l'intervallo è valido. In particolare, si possono utilizzare i seguenti elementi:

bysecond si possono definire una serie di secondi, all'interno di un minuto, in cui la regola viene considerata valida, i secondi sono compresi tra 0 e 59

byminute contiene la lista dei minuti all'interno dell'ora, anch'essi compresi tra 0 e 59

byhour indica la lista di ore della giornata, valori inseribili sono compresi tra 0 e 23

byday indica la lista di giorni della settimana in cui la regola viene considerata valida, i valori possibili possono essere tra MO, TU, WE, TH, FR, SA e SU (che corrispondono rispettivamente a lunedì, martedì, mercoledì, giovedì, venerdì, sabato e domenica)

bymonthday indica la lista giorni del mese in cui la regola è verificata

byyearday indica la lista di giorni dell'anno, i valori sono compresi da 1 a 366

byweekno indica la lista di quali settimane all'interno dell'anno i valori vanno da 1 a 53

bymonth indica la lista di mesi all'interno dell'anno in cui la regola viene considerata valida

Quando si inseriscono più valori all'interno di uno di questi campi, questi vanno separati da virgole. E' possibile inserire delle combinazioni degli elementi precedenti per ottenere delle regole estremamente complesse e articolate. Oltre agli attributi già visti ne esistono altri come wkst o bysetpos. E' possibile utilizzare l'output otherwise nel caso il momento di arrivo della chiamata non coincida con alcun intervallo di tempo presente all'interno dello switch.

Priority Switch

Vengono utilizzati per permettere agli script CPL di prendere decisioni in funzione della priorità specificata dal UA chiamante.

Lo switch è definito dal tag:

Tag: <priority-switch>

Output: priority, otherwise

L'output utilizzato è priority, e può utilizzare uno tra questi parametri: greater, less e equal, che rendono verificata la condizione rispettivamente se la priorità specificata è maggiore, minore o uguale al valore espresso. I valori possibili che possono essere utilizzati sono: emergency, urgent, normal e non-urgent, che corrispondono a priorità decrescenti. Nel caso non sia specificata alcuna priorità all'interno della chiamata, questa viene considerata di tipo normal.

2.1.2 Nodi di modifica delle destinazioni

Vengono utilizzate per modificare l'indirizzo originale a cui è indirizzata la chiamata, sono usati per reindirizzare le chiamate all'interno del dominio SIP (e non solo) per poter, ad esempio, avere all'interno del dominio degli indirizzi differenti da quelli visti dall'esterno. Può essere utilizzato per reindirizzare le chiamate, ad esempio, a seconda dell'ora del giorno, ad un centralino o ad una casella vocale. Funzionano in maniera diversa dai nodi switch in quanto (a parte un caso), non venendo effettuate delle scelte non sono presenti output, e quindi una volta eseguito il tag l'esecuzione passa direttamente al nodo successivo (cioè il nodo interno).

Location

Viene utilizzato per aggiungere esplicitamente un indirizzo alla lista attuale degli indirizzi per la chiamata.

Tag: <location>

Parametri: url, priority, clear

L'indirizzo da aggiungere viene specificato all'interno del parametro url secondo le regole definite dal protocollo sottostante. Un solo indirizzo può essere inserito in ogni nodo di tipo location, se si volessero aggiungere più indirizzi è sufficiente inserire più nodi di questo tipo in cascata.

Il parametro priority specifica un valore numerico (decimale) che va da 0.0 a 1.0, dove 1.0 è la massima priorità; viene utilizzato dal server per decidere l'ordine in cui eseguire le chiamate nel caso della presenza di più indirizzi. Nel caso non venga specificato viene preso in considerazione il valore 1.0.

Il parametro clear specifica se la lista dagli indirizzi già presente debba essere svuotata prima di aggiungere questo URL, i valori possibili sono yes o no. Specificando questo attributo automaticamente viene considerato questo indirizzo come la nuova destinazione della chiamata.

Lookup

Questo nodo viene utilizzato per modificare la lista delle destinazioni da una fonte esterna che dipende, non tanto dal protocollo utilizzato, ma dal sistema software che gestisce lo script CPL.

Questo tag ha le seguenti caratteristiche:

Tag: <lookup>

Output: success, notfound, failure

Parametri: source, timeout, use, ignore, clear

È presente un solo parametro obbligatorio, source, nel quale viene specificato l'URL che deve essere interrogato per ricevere la lista degli indirizzi da aggiungere.

In maniera simile al tag location è possibile specificare l'attributo clear per eliminare gli URL già presenti nella lista delle destinazioni.

Gli altri attributi sono specifici della fase di interrogazione, in particolare si ha timeout che specifica il numero di secondi che deve rimanere in attesa di una risposta, scaduti i quali si considera la ricerca infruttuosa.

Dato che non è garantito che la richiesta abbia esito positivo, all'interno di questo tag sono presenti tre output che vengono scelti a seconda del risultato dell'operazione: *success* se l'operazione è andata a buon fine e sono stati aggiunti degli indirizzi alla lista attuale

notfound se l'operazione è andata a buon fine, ma non sono stati ritornati indirizzi da aggiungere alla lista attuale

failure se ci sono stati errori nell'operazione di interrogazione, o se il timeout specificato è scaduto

Location Removal

Viene utilizzato per rimuovere indirizzi specifici dalla lista degli indirizzi, l'uso di questo nodo dipende fortemente dal protocollo sottostante e dall'applicazione che lo gestisce.

In particolare è descritto dal seguente tag:

Tag: <remove-location>

Parametri: location, param, value

L'argomento location specifica l'indirizzo o il pattern di indirizzi da rimuovere, nel caso non venga specificato vengono utilizzati gli altri parametri per specificare il comportamento di questo nodo. Gli altri due parametri, param e value, specificano, rispettivamente, una serie di nomi di parametri e della corrispondente serie di valori, separati da virgole, che vengono passati al sistema che gestisce lo script CPL per specificare le modalità di eliminazione degli indirizzi. I parametri dipendono dal sistema usato e dal protocollo sottostante.

2.1.3 Nodi di segnalazione

Servono per effettuare delle operazioni relative al protocollo sottostante, è possibile inoltrare le chiamate, reindirizzarle o mandare risposte arbitrarie al chiamante.

Proxy

Questo nodo inoltra la chiamata verso la lista delle destinazioni attualmente presente, il nodo ha le seguenti caratteristiche: Tag: <proxy>

Output: busy, noanswer, redirection, failure, default

Parametri: timeout, recurse, ordering

L'azione specifica intrapresa dal sistema quando deve eseguire questo nodo, dipende dal protocollo di segnalazione utilizzato; nel caso di SIP viene mandato un pacchetto di INVITE a tutti gli indirizzi presenti all'interno della lista delle destinazioni e si rimane in attesa di una risposta. Dopo che l'operazione di proxy termina, il server CPL sceglie la risposta "migliore" tra quelle ricevute, utilizzando i criteri definiti dal server o dall'amministratore del dominio. Se la chiamata avviene con successo lo script CPL termina, altrimenti viene eseguito un output a seconda della situazione che si viene a creare. In particolare gli output possibili sono:

busy l'indirizzo a cui viene inoltrata la chiamata non è in grado di rispondere

noanswer non c'è stata alcuna risposta in tempo utile

redirection la chiamata è stata rediretta ad un altro indirizzo

failure è stato ricevuto un messaggio di errore

default viene selezionato quando si verifica una situazione per cui non è stato specificato l'output

Se non viene specificato un output di tipo default e si verifica una situazione che non è gestita dagli altri output presenti, lo script termina. All'interno del tag proxy è possibile specificare il tempo, passato il quale, la chiamata viene considerata senza risposta attraverso il l'attributo timeout; se questo non viene specificato, il valore di base deve essere considerato di 20 secondi. Il parametro recurse (che può assumere i valori yes o no) serve per specificare se il server deve occuparsi automaticamente di effettuare un successivo inoltro della chiamata, nel caso che la risposta ricevuta dal destinatario sia di redirezione: in questo caso non verrà mai scelto l'output redirection in quanto viene gestito automaticamente dall'impostazione recurse. Il parametro ordering specifica come verranno eseguite le chiamate nel caso ci sia più di un indirizzo nella lista delle destinazioni. I valori possibili che può assumere sono:

parallel vengono eseguite tutte le chiamate contemporaneamente

sequential le chiamate vengono eseguite una dopo l'altra in sequenza seguendo il valore di priorità che è stato loro assegnato

first-only viene eseguita solamente la chiamata verso l'indirizzo con maggior priorità

Redirect

Questo nodo istruisce il server CPL di rispondere al chiamante di reindirizzare la chiamata trasmettendo la lista delle destinazioni attuale. Le caratteristiche di questo nodo sono:

Tag: <redirect>

Parametri: permanent

Appena viene eseguito, questo tag, termina l'esecuzione dello script CPL, di conseguenza questo è un nodo terminale, in quanto non può avere né output né nodi successivi. Il parametro permanent, che può assumere i valori yes o no, indica se la risposta al chiamante deve indicare se la redirectione è permanente o meno.

Reject

Quando viene incontrato questo nodo viene spedito un messaggio di rifiuto della chiamata a chi l'ha generata. Le caratteristiche di questo nodo sono:

Tag: <reject>

Parametri: status,reason

Anche questo nodo, appena viene eseguito, termina l'esecuzione dello script.

E' necessario specificare il codice del motivo del rifiuto attraverso il parametro status, i valori possibili sono:

busy il destinatario è occupato

notfound il destinatario non è stato trovato

reject la chiamata è stata rifiutata

error è stato generato un errore

E' possibile inserire come valore un codice specifico di un protocollo; oltre al codice dell'errore è possibile inviare anche una descrizione testuale del motivo attraverso il parametro reason.

2.1.4 Nodi generici

Oltre ai tipi di nodi già visti esistono altri nodi che effettuano altri tipi di operazioni.

Mail

Questo nodo notifica al server di informare l'utente dello stato dello script CPL attraverso una E-Mail. Le sue caratteristiche sono:

Tag: <mail>

Parametri: url

L'unico attributo che viene utilizzato è url, in cui va indicato l'indirizzo di posta elettronica destinazione come URL (specificando il protocollo mailto). E' possibile specificare altre informazioni, come l'oggetto della email, attraverso i parametri dell'URL.

All'interno della email deve essere descritto lo stato attuale dello script CPL e della chiamata che l'ha generata.

Log

Questo nodo richiede al server di memorizzare delle informazioni sulla chiamata all'interno di un registro, le sue caratteristiche sono:

Tag: <log>

Parametri: name,comment

Questo nodo accetta due argomenti, entrambi opzionali: name in cui viene specificato

il nome del registro, e comment in cui viene inserito un commento. Il server dovrebbe inoltre includere altre informazioni riguardo alla chiamata.

Sub

Questo nodo esegue un determinata procedura definita attraverso un tag subaction all'interno dello stesso script. Il tag è definito nel seguente modo:

Tag: <sub>

Parametri: ref

L'esecuzione viene passata alla subaction identificata con il nome che viene specificato all'interno del parametro ref. Questo nodo non può avere altri elementi CPL al suo interno, in quanto le operazioni che seguono si trovano all'interno della relativa subaction.

2.1.5 Altri nodi

Oltre ai tag principali incoming e outgoing, è possibili avere direttamente nell'elemento radice (cpl) dello script altri due tag, il primo subaction viene utilizzato per definire delle procedure personalizzate, il secondo ancillary viene utilizzato per definire delle estensioni del linguaggio CPL.

Subaction

Come già visto in precedenza esiste la possibilità di definire delle procedure che possono venir chiamate da qualunque punto dello script. Le caratteristiche di questo tag sono:

Tag: <subaction>

Parametri: id

Questi procedure hanno una limitazione: non possono essere richiamate più di una volta all'interno della stessa esecuzione dello script, questo per evitare la possibilità a di avere cicli di lunghezza indeterminata. Quindi è necessario, nel momento in cui lo script viene caricato nel server, che venga controllato che questo vincolo sia rispettato, ed in caso contrario rifiutare lo script.

Il parametro id viene utilizzato per definire il nome della procedura, che viene poi utilizzato all'interno del parametro ref del nodo sub, i nomi sono case-sensitive

Ancillary

Questo nodo è stato definito per poter aggiungere delle estensioni al linguaggio CPL, attualmente non è stato ancora utilizzato.

3 La libreria JCpl

La libreria JCPL è da considerarsi un Framework per lo sviluppo di call services (basati principalmente su SIP). Essa permette la creazione di servizi basati su CPL ed è uno strumento per l'interpretazione di un CPLs indipendentemente dal protocollo di segnalazione. La libreria JCPL è stata sviluppata in linguaggio java 1.5. Per la validazione degli script cpl è stata analizzata e utilizzata la libreria cpled.

■ Principali funzionalità

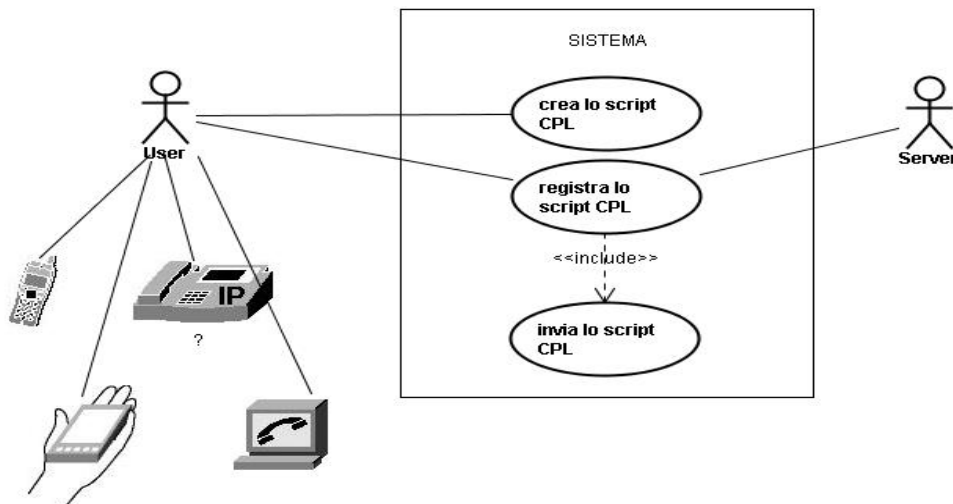
1. Gestione informazioni di una chiamata
2. Gestione degli script CPL (CPLs): salvataggio e lettura
3. Interpretazione delle istruzioni ed esecuzione di un CPLs

1 Gestione Informazioni di una chiamata

Sono stati implementati classi per interpretare le informazioni su una chiamata:

- Vengono impostati tramite i metodi di set della classe CPLCallInformation
- È stata inoltre sviluppata la classe CPLSipCallInformation che estende CPLCallInformation e che estrae le informazioni da una istanza della classe Message della libreria MjSIP. La realizzazione è stata complicata a causa dei numerosi tag richiesti nell'interpretazione

2 Gestione degli script CPL

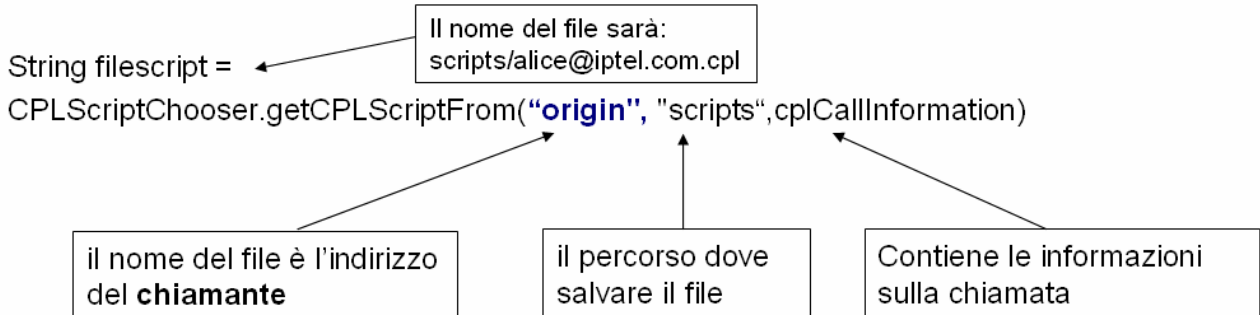


La libreria permette di gestire gli script Cpl

- Salvataggio di un CPLs di un end user nella fase register
- La libreria JCPL mette a disposizione una classe per ottenere il nome del file CPLs da salvare in una posizione desiderata

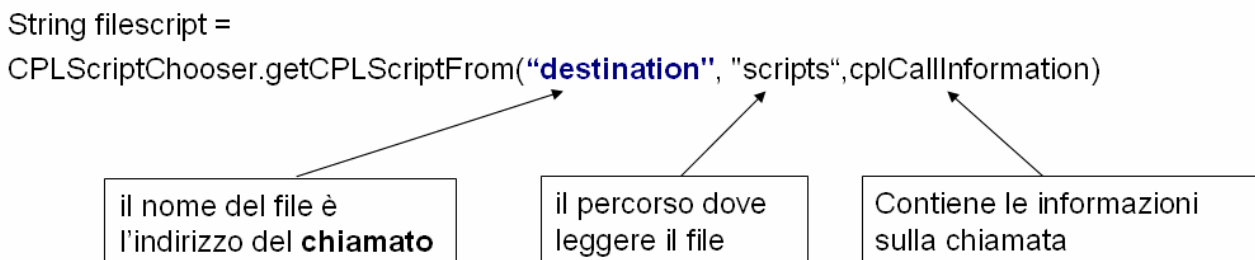
■ Salvataggio di un file CPLs

- Supponendo di aver estratto tutte le informazioni sulla chiamata (indirizzo chiamante) in `cplCallInformation` dal messaggio di Register



■ Lettura di un file CPLs

- Quale CPLs scegliere all'arrivo di un messaggio di INVITE?
- Supponendo di aver estratto tutte le informazioni sulla chiamata (indirizzo chiamante) in `cplCallInformation` dal messaggio di INVITE



3 Interpretazione delle istruzioni ed esecuzione di un CPLs

Per interpretare le diverse istruzioni si è suddiviso il problema in quattro moduli fondamentali.

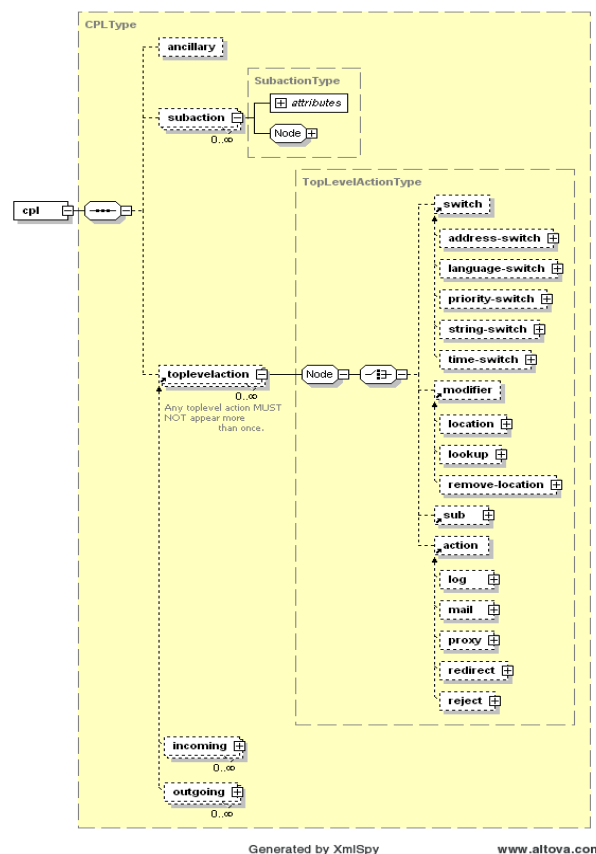
1. Parser di CPLs
2. Rappresentazione in classi java di CPLs
3. Istruzioni ed esecuzione
4. Interfaccia della azioni

1. Parser di CPLs

- Eseguire il parser di un file CPLs
- Parser Sax (event driven)
- Creare una rappresentazione in java della struttura ad albero di un CPLs
- Compire la validazione del CPLs

2 Rappresentazione in classi java di CPLs

- Collezioni di oggetti java organizzati seguendo la struttura ad albero tipica di un file xml, in questo caso di un file CPLs
- Ogni oggetto corrisponde ad un tag del file CPLs



3 Istruzioni ed esecuzione

- Le istruzioni permettono di attraversare l'albero della rappresentazione di CPLs
- Ad ogni istruzione è associato un oggetto della rappresentazione e l'informazione sulla chiamata
- In base alle informazioni che porta l'oggetto e la chiamata, una istruzione esegue un'azione
- In base al tipo di azione e all'esito della loro esecuzione, le istruzioni decidono quale sarà la prossima istruzione da eseguire e quale oggetto della rappresentazione passare (alla prossima istruzione)
- Attenzione: l'attraversamento dell'albero può avvenire solo in profondità

3.1 Le azioni si dividono in quattro categorie:

1. Azioni di SWITCH: scelta della prossima istruzione in base alle informazioni della chiamata
2. Azioni di SEGNALAZIONE: segnalazione e scelta della prossima istruzione in base all'esito dell'azione di segnalazione
3. Azioni di MODIFICA DELLE LOCAZIONI: modifica delle locazioni e scelta della prossima istruzione in base all'esito dell'azione di modifica
4. Azioni Varie: azioni di invio email e di log che non necessitano successive azioni di scelta

Importante

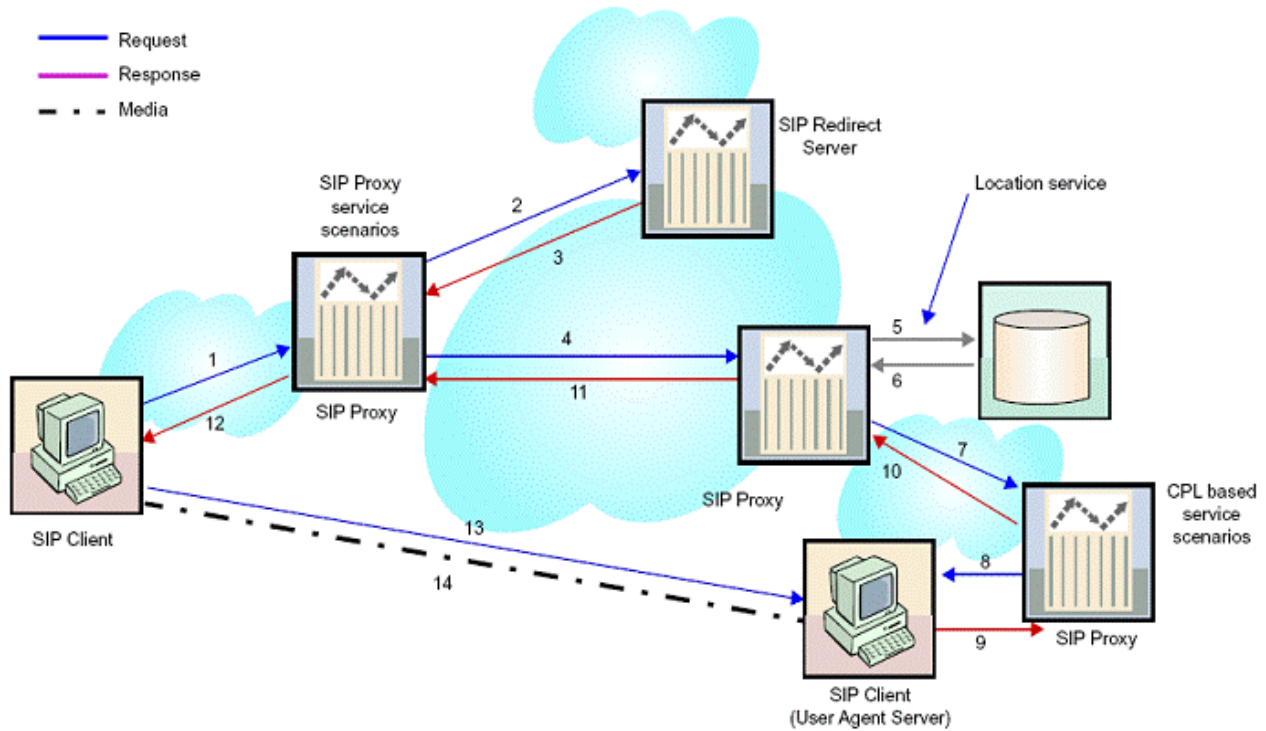
- Le azioni di SWITCH scelgono solo la prossima istruzione da eseguire (indipendenti dal protocollo di segnalazione e dalla libreria per la gestione delle locazioni)
- Il discorso è diverso per gli altri tre tipi di azioni

4 Interfaccia delle azioni

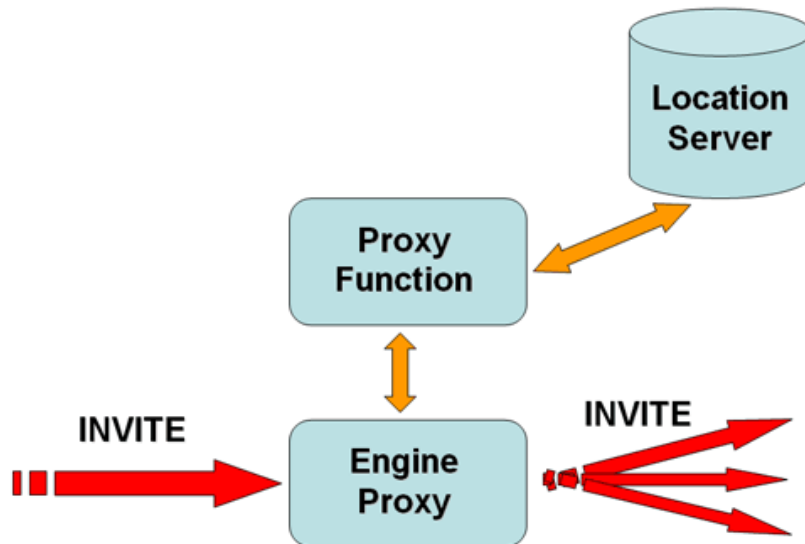
- Interfaccia per le azioni di segnalazione e modifica delle locazioni
- Per rendere la libreria indipendente dal protocollo di segnalazione dalle librerie per la gestione delle locazioni
- Implementazione delle azioni lasciate al server manager
- Libreria MODULARE

4 MjSIP con supporto CPL tramite JCPL

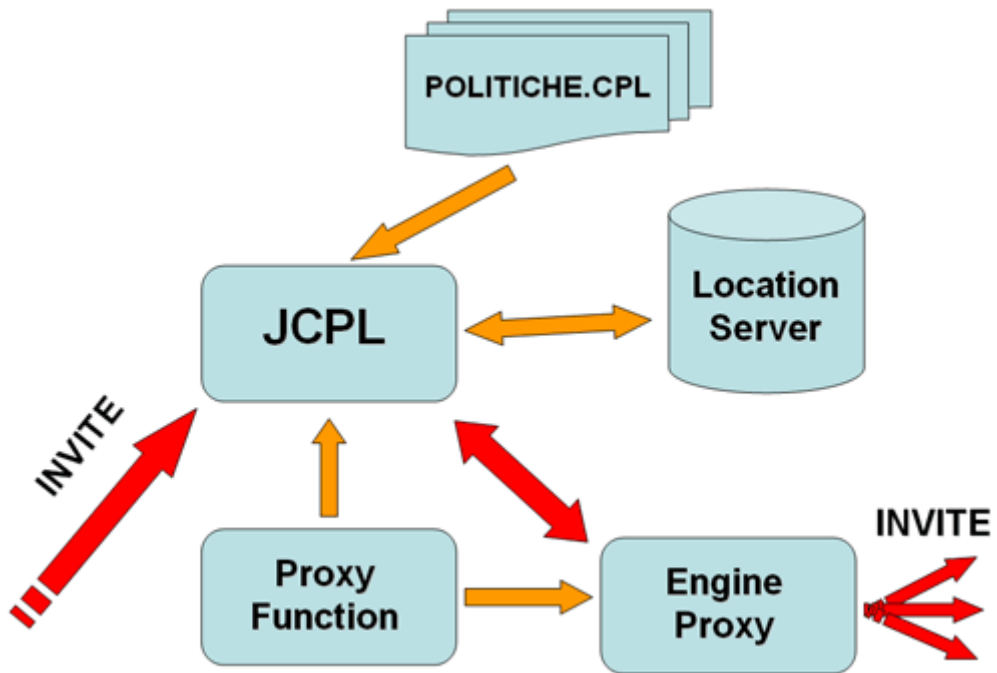
Un'iterazione generica tra 2 Client tramite protocollo SIP + CPL può essere così schematizzata:



Osservando solo la parte del server SIP e concentrandosi sull'applicativo MjSIP, il funzionamento logico è il seguente:



Di seguito è riportato il funzionamento di MjSIP con l'aggiunta del supporto a CPL tramite JCPL:



5 Caratteristiche dell'applicazione

La Libreria JCpl risulta essere SCALARE e prevede:

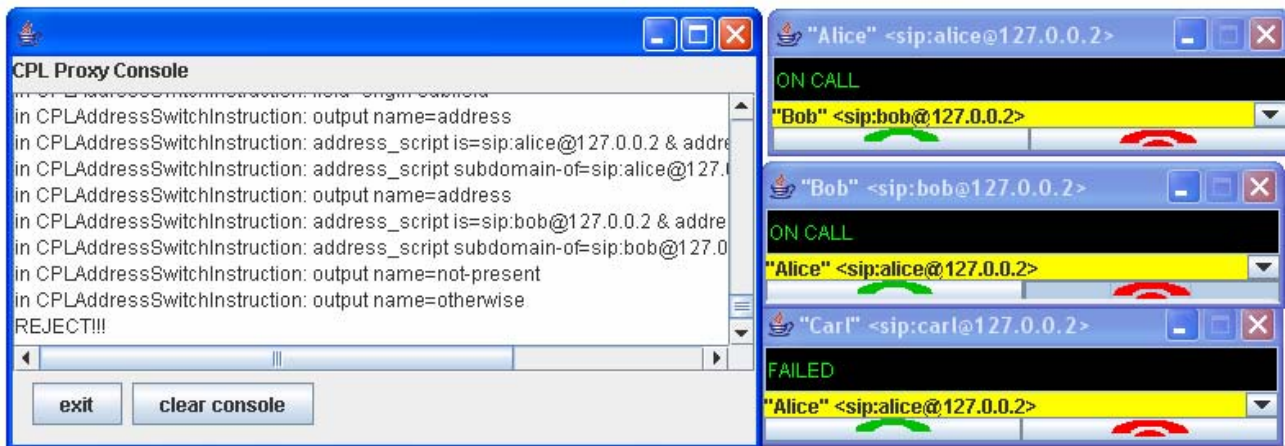
1. Possibilità di estendere la rappresentazione (discussa in precedenza)
2. Possibilità di aggiungere nuove istruzioni

La libreria JCPL è stata validata sviluppando un'applicazione che utilizza JCPL e MjSIP come libreria per la gestione della segnalazione e della modifica delle locazioni

E' stata sviluppata inoltre un' interfaccia grafica per rendere più *user friendly* il programma.

Elenchiamo alcune particolarità dell'applicazione

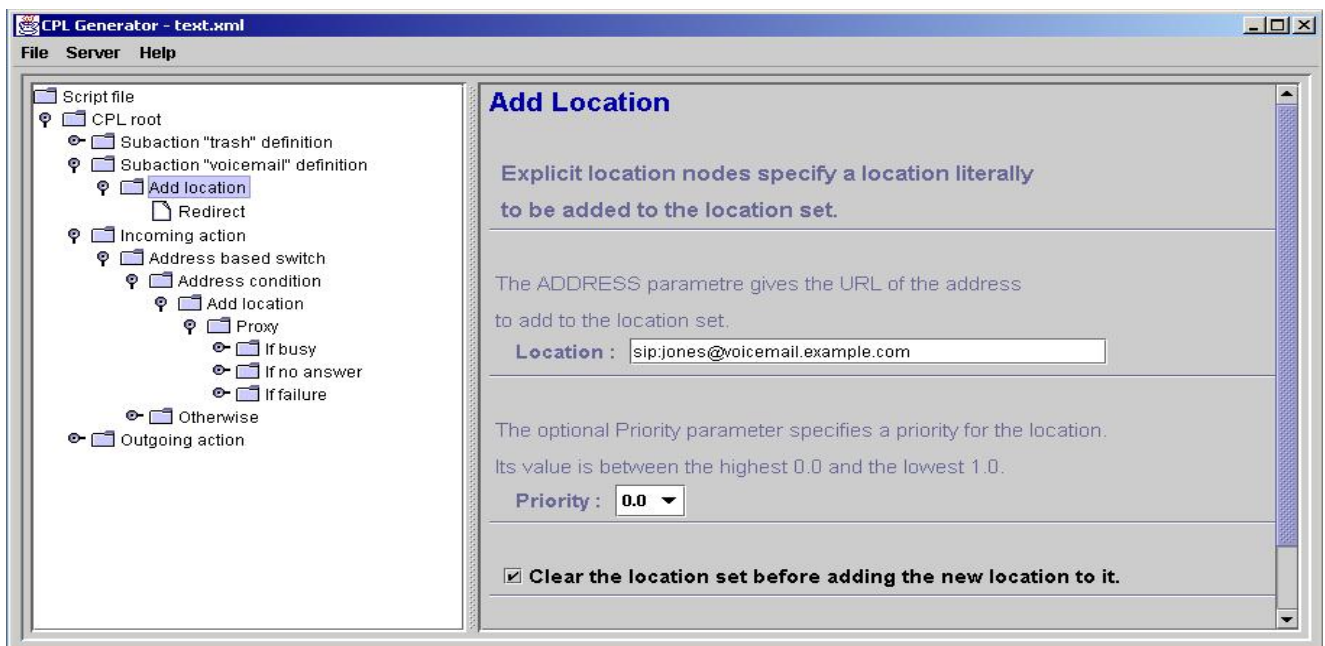
- Il progetto nella versione definitiva ha portato alla creazione di circa 120 file java
- E' stato reso indipendente dalla piattaforma e modulare
- Sono stati creati file eseguibili delle singola applicazioni per windows e per linux
- Sono state implementate numerose librerie
- Rilevati e corretti bug software
- Tutte le transizioni vengono salvate in file di log e visibili all'utente attraverso interfaccia grafica



6 CplEditor

Per la validazione dei file cpl è stata studiata e sfruttata la libreria CplEdit. Esso è un software grafico in java per effettuare il parser di file xml in script cpl e scaricabile da <https://sourceforge.net/projects/cpled/>

- **E' possibile modificare script cpl aggiungendo tag specifici tutto attraverso interfaccia grafica**
- **L'applicazione è stata compilata e adattata al progetto jcpl. E ' inoltre stato creato un eseguibile per sistemi operativi windows.**



Per il funzionamento specifico della libreria JCpl vedere la documentaione *jcpl per esempi*

7 Il Software

- Tutto il software sviluppato è fornito su supporto digitale (consegna CD)
- E' stato creato un cd con avvio automatico e software per l'installazione (sotto windows) del progetto



■ Vediamo la struttura delle cartelle:



- Nella cartella JCPL sono contenuti i sorgenti della libreria JCPL
- Nella cartella CPLApp_Win sono contenuti i sorgenti dell'applicazione che utilizza JCPL e Mjsip (applicazione sviluppata per validare JCPL e dimostrarne l'utilizzo)
- Nella cartella CPLApp_Linux sono contenuti i sorgenti eseguibili sotto distribuzioni Linux.
- La sotto cartella dist, presente in entrambe le cartelle, contiene rispettivamente le due applicazioni compilate (file jar)
- Nella cartella Cpled_Win editor grafico cpl per la validazione dei file

Riferimenti Bibliografici:

- A) Sito Mjsip (<http://www.mjsip.org>)
- B) RFC 3880
- C) RFC 2824
- D) RFC 3261
- E) RFC 3066
- F) RFC 2445
- G) H.M. Deitel P.J.Deitel "Tecniche avanzate di programmazione" "terza edizione"
- H) Steven John Metsker "Design Pattern in Java" "Addison Wesley"