ANALISI E PROGETTAZIONE DI UNA ARCHITETTURA PER L'HOME COMPUTING BASATA SU XML

Iacopo Iacopini, Luca Tarrini, Vittorio Miori, Rolando Bianchi Bandinelli

INDICE

1. DOMOTICA

Gli ultimi decenni del secolo scorso hanno cambiato, anche talvolta sostanzialmente, il nostro modo di vivere e di pensare l'ambiente domestico. Un cambiamento dovuto alla necessità comune di migliorare l'accessibilità dell'ambiente, l'abitabilità, il comfort, la sicurezza e la qualità della nostra vita. Per questo sono entrati a far parte delle nostre case elettrodomestici sempre più sofisticati e all'avanguardia, dotati di "intelligenza". Con questo non dobbiamo pensare solo al Personal Computer, ormai presente in quasi tutte le case, ma anche a TV color sempre più complessi, lettori DVD, VCR, stereo superaccessoriati, forni a microonde, piani cottura, lavastoviglie e lavatrici dove la componente elettronica ha il sopravvento su quella meccanica.

Il grande passo che però solo negli ultimi anni si sta cercando di compiere è di interconnettere tutti questi dispositivi insieme a formare così un'unica rete grazie alla quale sia possibile gestire ogni apparecchiatura con facilità ed in modo standard ovunque sia necessario, e che dia la possibilità di far dialogare tra loro i componenti della rete.

La casa si sta avviando verso nuove frontiere di sviluppo e di automazione e la Domotica nasce proprio per venire in contro e soddisfare queste esigenze.

1.1. Cosa è la Domotica

Il termine Domotica deriva dall'importazione di un neologismo francese *domotique*, a sua volta contrazione della parola greca *domos* e di *informatique* e rappresenta la disciplina che si occupa dell'integrazione dei dispositivi elettronici, degli elettrodomestici e dei sistemi di comunicazione e di controllo che si trovano nelle nostre abitazioni. Le origini risalgono intorno agli anni 70 quando si vennero a sviluppare i primi progetti concreti di automatismi legati alla gestione di impianti di allarme o altre funzionalità come l'accensione, lo spegnimento e la temporizzazione delle luci.
La continua evoluzione delle tecnologie e lo studio più approfondito delle esigenze dei consumatori permettono di concepire questa disciplina lontano dall'idea della semplice informatica applicata alla casa. Domotica vuol dire interazione fra la casa e l'uomo, vuol dire ricerca di una migliore accessibilità e fruibilità dell'abitazione, vuol dire anche creare nuovi mezzi per condividere gli ambienti domestici con gli altri membri della famiglia e dare la possibilità a chi non ce l'ha di guadagnarsi l'indipendenza nella vita di tutti i giorni.

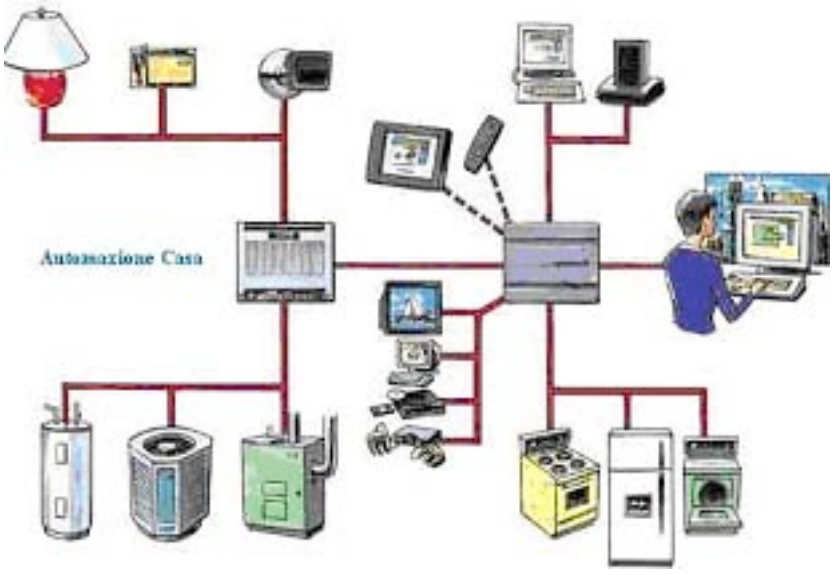


Figura 1: Nuovo concetto di "smart house" o "casa intelligente"

1.2. Tecnologie ed aspettative per il futuro

Lo scenario in cui viviamo vede una crescita costante e continua dell'uso della tecnologia all'interno degli ambienti domestici e di lavoro. Elettrodomestici ed apparecchi elettronici "intelligenti" ogni giorno vanno a sostituire i sempre più obsoleti e tradizionali dispositivi.
Merloni Elettrodomestici ad esempio ha da poco messo sul mercato un frigorifero, chiamato Opera, capace di dialogare con l'utente attraverso un semplice display in modo da comunicare la scadenza dei cibi, la mancanza di un prodotto e capace di autoprogrammare il suo funzionamento a seconda del contenuto. *Bang & Olufsen* ha prodotto un nuovo tipo di impianto audio con effetto surround per uso domestico capace di riprodurre fedelmente l'effetto cinema. Della *Palm* invece gli ultimi palmari capaci di navigare su internet e dotati di caratteristiche tecniche eccezionali abbinati a maneggevolezza e design accurati.
Ma questi sono solo alcuni esempi di quello che appare come un settore di mercato tutto in espansione che vede sempre più agguerrite sfide tra i costruttori. La casa di produzione *Ariston* sta sviluppando una serie di elettrodomestici chiamati Ariston digitel capaci di parlarsi e di scambiarsi informazioni tramite la rete elettrica in modo da ottimizzare e gestire al meglio i consumi energetici, e che comunicano direttamente al tecnico eventuali guasti, malfunzionamenti e usura delle loro componenti.
Le aspettative future puntano anche a immettere sul mercato degli elettrodomestici per i quali si potranno stipulare abbonamenti del tipo Pay-per-Use, che danno la possibilità, ad esempio, di affittare una lavatrice di ultimissima generazione e pagarne soltanto l'uso effettivo.
La *BTcino* ha messo appunto un sistema di domotica e teledomotica per il controllo a distanza dell'abitazione chiamato My Home. Questo è formato da una serie di dispositivi capaci di migliorare il comfort della casa e gestire la sicurezza delle cose ed delle persone, tramite un meccanismo di controllo da remoto di tutte le funzionalità degli elettrodomestici o degli impianti di allarme. Le interfacce di controllo possono essere di varie tipologie come semplici cellulari, palmari o PC portatili.
Come possiamo aver capito uno degli aspetti centrali di questo nuovo modo di vivere la casa è quello di vedere non più ogni elettrodomestico o ogni servizio dell'abitazione come unico ed isolato dagli altri, ci stiamo lentamente dirigendo verso una integrazione ed una coesistenza. Grazie all'uso di nuove tecnologie e nuovi standard potranno nascere anche nuove tipologie di servizi avanzati per gli utenti. L'utente disporrà di tutti i mezzi necessari per tenere sotto controllo il sistema abitazione, e saranno i singoli dispositivi a gestire tutto il lavoro e a fornire in modo automatico i servizi, decidendo tra loro priorità, privilegi, modalità di uso ed eventuali collaborazioni.

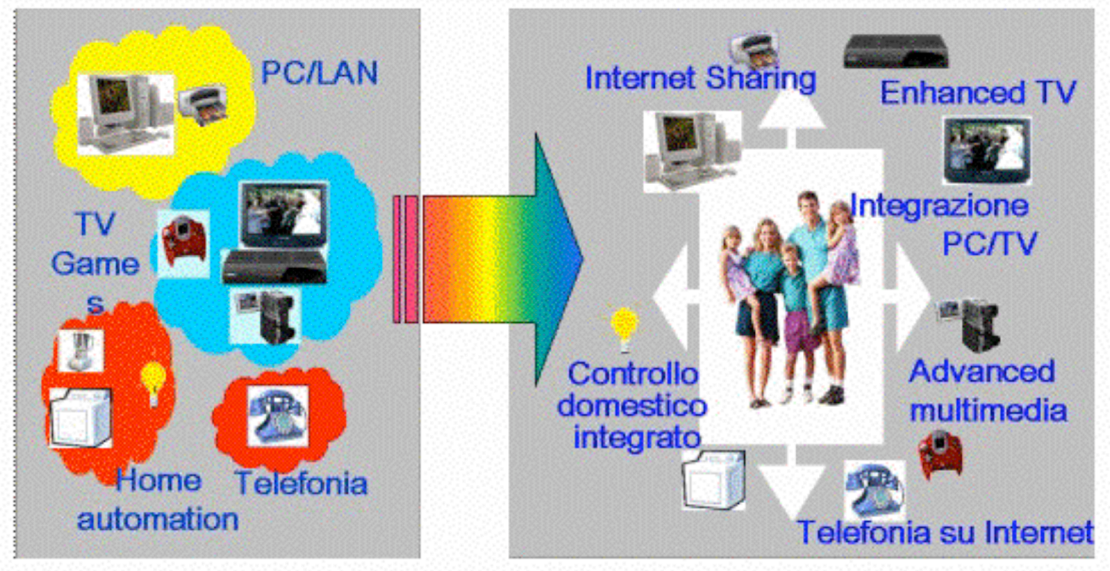


Figura 2: L'integrazione sarà alla base delle future tecnologie

La Domotica all'interno di questo sistema avrà il compito di pianificare e di gestire tutti i dispositivi ad uso domestico e di integrarli con i sistemi di comunicazione e di controllo a distanza. Troveremo a nostra disposizioni sistemi centralizzati di controllo dell'abitazione ai quali impartire comandi più o meno semplici anche attraverso l'attivazione vocale.

1.3. Il problema dell'interoperabilità

Uno dei traguardi principali della Domotica è quello di riuscire ad interconnettere tra loro ogni dispositivo all'interno dell'abitazione di qualunque tipo o fatture esso sia. In diretta conseguenza nasce la necessità di connettere tutta la rete domestica al mondo esterno, e quindi alle reti di pubblico dominio, così da permettere agli operatori di telecomunicazione di fornire servizi avanzati e agli utenti domestici di usufruirne. Un altro aspetto molto importante è quello del controllo, della manutenzione, e della gestione delle risorse a distanza.
Tutto questo porta alla necessità di avere un apparato capace di permettere alle varie reti di interagire e connettersi bidirezionalmente con la rete pubblica. Esso non sarà esattamente un PC, ma un qualcosa più simile ad un router con funzionalità avanzate, a basso costo e con architettura flessibile in modo da garantire l'adattamento fra reti indipendenti e diverse tra loro. Queste sono le funzionalità che dovrà implementare un così detto "Residential Gateway", le cui caratteristiche, oggi allo studio di numerosi gruppi di ricerca e iniziative di standardizzazione, mirano a farlo diventare parte integrante dei servizi e delle funzionalità delle Home & Office Network con un ruolo di privilegio rispetto alle altre periferiche.
Oggi le vie che portano verso lo sviluppo di un Residential Gateway sono molteplici, alcuni cercano di creare un dispositivo unico che abbia un ruolo centrale all'interno della rete e che gestisca da solo tutto il carico di lavoro. Altri puntano invece a creare un sistema distribuito con tutti i componenti sincronizzati e comunicanti con gli uni con gli altri.
I problemi legati alla sua realizzazione sono molteplici, infatti, dovrà rispettare contemporaneamente requisiti come affidabilità, sicurezza, semplicità di uso, costi ridotti e dovrà supportare l'esecuzione di più processi in contemporanea. I principi base di versatilità, flessibilità e la gestione di più processi sono sicuramente gli obiettivi più onerosi e difficili da raggiungere in quanto dovrà essere garantita l'interconnessione di dispositivi del tutto diversi tra loro, magari residenti su reti diverse e di diversa concezione, dovrà fornire servizi, e dovrà gestire anche un ruolo di interfaccia con le reti esterne.
Nasceranno senza dubbio problematiche legate alla sicurezza non più dell'ambiente domestico, ma dello stesso impianto di gestione e controllo. Sarà inoltre difficile stabilire confini fisici di proprietà e di gestione delle responsabilità all'interno delle reti.
L'abitazione è oggi raggiunta da una grande quantità di servizi provenienti da sistemi di produzione e da fornitori indipendenti, incompatibili tra loro, che si differenziano anche per il supporto fisico di trasmissione: cavi, doppini telefonici, etere, linea elettrica. E' quindi molto importante iniziare a pensare ad una via per gestire in maniera unitaria un apparato che rischia di diventare troppo complesso. Fra le industrie del settore c'è un sempre più crescente interesse verso una soluzione che interfacci tra loro i diversi segnali provenienti dall'esterno verso l'abitazione e permetta il dialogo fra i vari sottosistemi.

1.4. Standard tecnologici di riferimento

Moltissimi sono gli standard utilizzati nel campo dell'automazione domestica, e forse questo è anche uno dei motivi che hanno rallentato la crescita di questo settore e che hanno fatto crescere un'incertezza sia da parte del consumatore che da parte delle stesse case produttrici.
Ma vediamo una panoramica sui principali suddivisi per categorie.

1.4.1. Standard per la comunicazione

- *All Bus Datapark*
Tecnologia basata sulla trasmissione bidirezionale impulsiva in banda base su una delle quattro dorsali bus a disposizione. Delle quattro dorsali chiamate Databus, Parkbus, widebus e Virtualbus solo le prime tre sono costituite da un monofilo telefonico riferito a terra, mentre la quarta non necessita di un vettore fisico (onde convogliate). La loro banda passante è molto grande, infatti spazia da 3Hz a 5/6 MHz, e questo da la possibilità di trasmettere dati sia in analogico che in digitale.

- *Bati Bus*
Questo standard fa uso dei normali doppini telefonici per permettere la comunicazione di comandi tra CPU, attuatori e sensori all'interno di una rete di automazione domestica. Nel 1999 è stato una delle basi su cui sviluppare il più famoso standard KNX (konnex).

- *Bluetooth*
Fondato nel 1998 da colossi come Ericsson, IBM, Intel, Nokia e Toshiba, utilizza un sistema radio che opera sui 2,4 GHz e permette ad apparecchiature elettroniche ed informatiche di dialogare tra loro con velocità fino ad 1 Mbps in un raggio massimo di alcune decine di metri.

- EIB (European Installation Bus)
E' uno standard aperto, disponibile a tutti i costruttori che intendono offrire soluzioni per l'automazione domestica o degli uffici. Può essere adottato su molti tipi di linea come il doppino, la powerline, ethernet, radiofrequenza ed infrarosso. Si è diffuso molto sul mercato ed è andato poi a confluire nello standard KNX.

- EHS (European Home System)
Nato con la collaborazione fra case costruttrici e agenzie governative è nato per garantire la comunicazione fra apparecchiature presenti all'interno di una rete domestica o dell'ufficio. Supporta milioni di indirizzi, comprende la funzionalità Plug & Play e dispone di un efficace sistema di gestione degli errori che lo rende particolarmente affidabile. Anche questo progetto è andato a confluire in KNX.

- Ethernet
Noto anche come IEEE 802.3 è stato alla base dello sviluppo delle reti locali degli ultimi anni. Il costo ridotto della tecnologia dovuta alla semplicità del suo standard e la possibilità di sfruttare il doppino telefonico per velocità di trasmissione comprese tra 1 Mbps e 1 Gbps sono il suo punto di forza e lo rendono il protocollo in assoluto più utilizzato. Negli ultimi tempi si cerca di utilizzarlo anche per l'automazione domestica.

- *HBS (Home Bus System)*
Nato nel 1988 in risposta da parte di alcune casa giapponesi agli standard europei e americani, utilizza due cavi coassiali e otto coppie di twisted-pair a cui vengono collegati tutti gli apparecchi audio/video, telefoni e altri dispositivi.

- *No New Wires*
Permette la realizzazione di una rete multimediale ad alta velocità su rete elettrica, che può essere integrata con collegamenti radio per coprire anche i punti dove non è disponibile la cablatura elettrica. Può raggiungere velocità fino a 14 Mbps.

- *Shareware*
Membro della Wireless Ethernet Compatibility alliance (WECA) offre soluzioni hardware e software che permettono connessioni con alte prestazioni fra i dispositivi all'interno dell'abitazione.

- *X-10*
Presente da moltissimo tempo sul mercato statunitense e diffusosi anche in Europa è diventato uno standard storico e consolidato, anche grazie alla innumerevole presenza di dispositivi sul mercato. E' costituito da una unità centrale che invia i comandi ai dispositivi periferici ed sfrutta come linea di trasmissione la rete elettrica.

1.4.2. Standard per la costruzione di apparati domotici

- Cebus (Consumer Elettronics Bus)
Sviluppato dall'americana Electronic Industries Association, è uno standard integrato multimediale per sistemi di Home Automation che ha come caratteristiche principali la flessibilità e la modularità. Lo svantaggio legato a questo è che i dispositivi che lo supportano necessitano di sufficiente potenza di elaborazione per permettere la trasmissione dei dati.

- HAVI (Home Audio Video Interoperability)
Sviluppata da otto grandi case produttrici di apparecchiature elettroniche di consumo come Grunding, hitachi, Matsushita, Philips, Sharp, sony, Thompson e Toshiba con l'architettura di una rete domestica paritetica e distribuita. E' basata sull'interfaccia i.LINK ed offre servizi come la connettività Plug & Play, l'interoperabilità tra apparecchiature di marche diverse e facilità d'uso.

- *HES (Home Electronic System)*
E' uno standard internazionale sviluppato da esperti con varia nazionalità e provenienza. Gli esperti sono organizzati in un gruppo di lavoro noto come ISO/IECJTC1/SC25/WG1 che ha il compito di sviluppare standard e sottoporli alle nazioni partecipanti per l'approvazione. Gli obiettivi prefissi sono quelli di un interfaccia universale, un linguaggio di comunicazione standard e un Residential Gateway che permetta la comunicazione con l'esterno.

- *Home Plug and Paly*
Le sue specifiche che rientrano in quelle definite dal CEBus regolano l'interazione ad alto livello tra dispositivi ed applicazioni. Questo comporta che ogni costruttore possa progettare i suoi apparecchi senza conoscere niente del funzionamento delle altre apparecchiature con cui vorrà dialogare.

- *Jini*
La missione di questo standard basato su tecnologia Java è quella di permettere la nascita di un ampio numero di dispositivi consumer interoperabili tra loro, mediante uno standard industriale aperto. I questo modo si da la possibilità di effettuare scambio di dati togliendo però l'onere alle case costruttrici di pensare anche al mezzo fisico di comunicazione.

- *KNX (Konnex)*
E' nato in seguito ad una confluenza in questo unico progetto di 3 standard come EIB, BATIBUS, EHSA in modo da promuovere uno standard unico dedicato al home e building automation. Lo standard include in se tutte le principali caratteristiche dei tre da cui deriva. I componenti, realizzati da costruttori diversi, vengono garantiti, dopo una certificazione della Konex, per essere interoperabili e funzionare correttamente senza il bisogno di interfacce. Supporta varie modalità di

funzionamento e diversi mezzi trasmissivi ed è stato pensato per diventare un marchio di qualità per il consumatore.

- *LonWorks*
Creata più di 10 anni fa dalla Echelon Corporation costituisce una piattaforma multimediale, aperta ed indipendente dai tipi di media trasmessi, per la gestione di dispositivi connessi in rete. Le sue specifiche possono essere utilizzate da chiunque su qualunque sistema senza il riconoscimento di royalities, e dopo una fase di accurati test il prodotto può utilizzare il logo LonWorks. Ne esiste attualmente una implementazione su chip che ha ottenuto un grandissimo successo sul mercato.

- OSGi (Open Service Gateway Initiative)
E' uno standard nato dalla collaborazione di 14 colossi dell'informatica, dell'elettronica e delle telecomunicazioni, basato su tecnologia JAVA è nato per interfacciare apparecchi domestici intelligenti con reti di dati. Comprendono una vasta gamma di servizi possibili come connettività internet, servizi TV intelligenti, funzioni di automazione domestica e monitoraggio a distanza.

- UPnP (Universal Plug and Play)
Utilizzando tecnologie e protocolli web standard permette ad un'ampia gamma di dispositivi di riconoscersi e di comunicare tra di loro mediante apparecchiature intermedie come PC o set-top box. Le sue specifiche sono fornite in modo aperto dal UPnP Forum e rappresentano delle linee guida da far rispettare ai singoli costruttori.

- *VESA Home Network*
Consente lo scambio di informazioni all'interno di apparecchiature digitali connesse all'abitazione e fornisce l'interoperabilità anche tra dispositivi presenti su reti diverse. Consente un controllo diretto sui dispositivi sia da parte dell'utente che dagli altri dispositivi e fornisce anche un'interfaccia comune fra i vari Residential Gateway presenti.

2. RESIDENTIAL GATEWAY (RG)

Una delle maggiori conseguenze della deregolamentazione e della mancanza di standard adeguati nel mondo dell'home automation è la competizione instauratasi fra i vari servizi, consolidati o emergenti, volti ad entrare nelle nostre case, come connessioni internet dial-up o high-speed, trasferimento di video in digitale, gestione della sicurezza, gestione dell'energia, telefonia IP e molti altri ancora. Questo comporta per gli utenti il dover spesso scegliere fra l'uno o l'altro servizio. Con l'aumento dei dispositivi presenti all'interno della casa, si assiste anche ad un esponenziale aumento della complessità di interconnessione all'interno della rete domestica. Tutto questo genera la necessità di un dispositivo capace di facilitare interscambi di informazioni e gli accessi alle singole periferiche.

La tendenza che si è riscontrata negli ultimi anni è stata ed è quella di cercare un modo di standardizzare le interfacce con le quali i singoli dispositivi si connettono alla rete, rimandando poi l'onere di supportare tutti i maggiori protocolli di comunicazione e di fornitura de servizi ad un altro dispositivo che gestisca la comunicazione sulla rete e si interfacci con le reti esterne. Questo è il ruolo che dovrà avere il Residential Gateway.

L'interesse nei confronti nell'Home Networking sta crescendo molto rapidamente anche dal punto di vista degli stessi fornitori di servizi e degli operatori di telecomunicazione che tendono a diventare *fornitori integrati* di servizi avanzati e nuove soluzioni di rete in grado di raggiungere nuove aree domestiche. Quello che però allo stesso tempo preoccupa è l'enorme varietà di specifiche di middleware, di iniziative e di consorzi di standardizzazione nel settore delle reti domestiche, ognuna delle quale tendenzialmente dedicata ad applicazioni specifiche ed a strati diversi del modello OSI (Open System Interconnection). Altro punto debole è la mancata chiarezza sul ruolo che dovrà avere l'IP all'interno delle Home & Office Network.

Da tutto questo ne deriva, appunto, la necessità di un middleware avanzato, per l'integrazione di applicazioni, che possa garantire la fornitura di soluzioni integrate.

2.1. Cosa sono i Residential Gateway

Il concetto di Residential Gateway (RG) è relativamente nuovo, venne associato per la prima volta al significato oggi riconosciuto soltanto nel 1995. La definizione più generica e forse quella in assoluto più utilizzata per il RG, altrimenti conosciuto anche come Middleware, è quella che lo descrive come "**un dispositivo di interfaccia con la rete che fornisce le risorse per accedere ad un servizio fornito all'abitazione come la telefonia, TV via cavo e servizi Internet/On Line**".

Più conciso ma forse anche più forte può essere descriverlo semplicemente come "**quel dispositivo che connette una Home Area Network ad Internet".**

Qualunque delle due definizioni si preferisca usare bisogna assolutamente tenere presente di non far confusione con altri tipi di dispositivi non generici ma costruiti con fini particolari come possono essere modem (apparecchi per la fornitura di connessioni internet) o set-top boxes (dispositivi per l'allacciamento a line di fornitura di servizi specifici come l'approvvigionamento energetico). Confusione che invece veniva inevitabilmente fatta prima che fosse presentata al pubblico nel 1995 una relazione sulla quale si faceva chiarezza e si davano definizioni precise circa applicazioni e sviluppi di questo dispositivo.

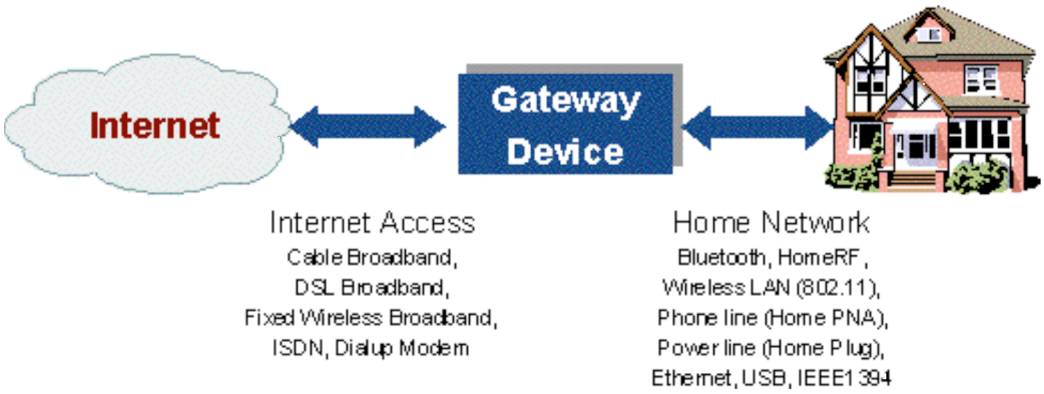


Figura 3: Scenario di uso del Residential Gateway

2.2. Le origini del Residential Gateway

Per decine di anni le aziende fornitrici di servizi di telefonia, di energia elettrica, televisivi ed altro ancora hanno sempre lavorato in un ottica di completa separazione l'una dalle altre. Questo è stato un fenomeno dovuto a regole che imponevano una netta distinzione e alla mancanza di tecnologie adeguate. Oggi si cerca invece di aggregare il più possibile i vari servizi in modo da abbattere i costi e la complessità di sistemi troppo vasti, allo stesso tempo sorge la necessità di creare un sistema capace di dare accesso ai servizi di diversi fornitori con flessibilità e libera scelta.

Il concetto associato al significato attuale di Residential Gateway è stato introdotto a partire dal 1995 quando l'RG Group nell'Ottobre del 1995, rilasciò un white paper intitolato "The Residential Gateway". Essenzialmente in questa relazione si usava il termine RG per rappresentare una interfaccia intelligente e centralizzata tra la rete esterna e quella domestica, inoltre gli si attribuivano due funzioni chiave:

- Quella di interfaccia fisica per i dispositivi con la rete esterna e di Termination Point per i fornitori di servizi che vogliono connettersi alla rete interna;
- Quella di rendere possibile la fornitura di servizi nuovi e consolidati al consumatore.

Essenzialmente *l'RG Group* immaginò il Residential Gateway come "un'interfaccia di rete singola, intelligente, standardizzata e flessibile capace di ricevere segnali di comunicazione da varie reti esterne e di smistarli attraverso la rete domestica ai singoli specifici dispositivi". L'idea base da sviluppare era quella di fornire agli stakeholders (fornitori, providers, clienti, software houses, etc.) i benefici dovuti ad un aumento della concorrenza e all'introduzione di nuove tecnologie e nuovi concetti di servizi avanzati.

2.3. Applicazioni possibili grazie ai RG

Il dispositivo pensato dall'*RG Group* rende possibile tutte le maggiori applicazioni che possono essere utilizzate nell'abitazione come:

- Telecomunicazione
- Accesso Internet
- E-Learning
- Telemedicina
- Videotelefono
- Gestione di applicazioni domestiche
- Gestione dei sistemi di sicurezza
- Lettura automatizzata dei consumi e dei contatori
- Video intercomunicazione

- VCR virtuali e video-on-demand
- CD jukebox
- Pubblicità On Line e cataloghi elettronici

Altri dispositivi più semplici sono nati per la gestione di videotelefonia e scambio di dati, e possono gestire contemporaneamente segnali vocali, video e dati smistandoli verso gli appropriati end point della rete domestica.

Purtroppo oggi il mercato fornisce molti servizi dedicati che gestiscono applicazioni specifiche o particolari categorie di applicazioni come accessi internet, telefonia, gestione dell'energia e gestione della sicurezza. Si ipotizza, comunque, che in pochi anni saranno sviluppati nuovi RG con caratteristiche più universali.

2.4. Benefici dovuti ai RGs

Benefici chiave forniti dai RG	
Stakeholders	**Benefici**
Utenti	1. *Accesso facilitato a molteplici tipologie di rete*: il RG fornisce agli utenti finali un'interfaccia amichevole che da la possibilità di accedere in modo semplice e veloce ad una vasta gamma di servizi e dispositivi, indipendentemente dalla loro complessità. 2. *Prezzo dei servizi più basso*: Avere un unico dispositivo standard da la possibilità all'utente di scegliere con più facilità fra le varie proposte del mercato che può espandersi con più facilità e quindi aumentare il numero delle offerte. 3. *Incremento dei servizi*: Il RG per sua natura apre la strada ad una innumerevole quantità di nuovi servizi sviluppabili, senza per questo ledere alla complessità del sistema. 4. *Acquisto di pacchetti di servizi*: Il RG da la possibilità agli utenti di acquistare con un unico pacchetto molteplici servizi integrati tra loro e offerte.
Fornitori di Servizi	1. *Riduzione dei costi*: Creare una piattaforma standard per tutti gli utenti può far abbassare notevolmente i prezzi di progettazione ed allo stesso tempo aumentare i servizi disponibili in quanto si passa ad economie su scala molto più ampia e a bacini di utenza molto vasti. 2. *Possibilità di aggiornare le tecnologie*: Si può pensare ad un aggiornamento automatizzato dei dispositivi e delle applicazioni domestiche in modo da fornire migliore qualità di servizio e migliore efficienza. 3. *Maggiore profitto*: Aumentando la quantità e la qualità dei servizi e semplificando la possibilità di offrirli le aziende sono portate ad ampliare le proprie vedute e le proprie entrate finanziarie. 4. *Gestione semplificata della rete*: Grazie al RG la gestione della sicurezza e la manutenzioni di guasti e malfunzionamenti della rete potrà essere del tutto automatizzata e gestita in remoto da postazioni fisse.
Costruttori	1. *Interfacce standard*: Grazie alla standardizzazione del RG che gestirà le interconnessioni all'interno della rete domestica i costruttori potranno sviluppare nuovi prodotti capaci di sfruttare a pieno l'interazione tra dispositivi. 2. *Riduzione dei costi di progetto e produzione*: La standardizzazione toglie ai costruttori la necessità di progettare le varie interfacce con le quali i singoli dispositivi dovrebbero comunicare con l'esterno e con gli utenti. 3. *Collaborazione*: La costruzione di ambienti intercomunicanti faciliterà l'interazione fra costruttori di differenti dispositivi che potranno dialogare e scambiarsi informazioni, aumentando in modo

	sostanziale le possibilità di fornire nuovi, migliori e più accurati servizi.

In un mondo in cui convenienza e flessibilità viaggiano di pari passo lo sviluppo di un dispositivo di interfaccia come il RG è da ritenersi indispensabile e non solo conveniente.

2.5. Industrie coinvolte nell'uso dei RG

Sono molte le categorie che direttamente o indirettamente potranno trarre vantaggi dall'uso dei Residential Gateway e che potranno sviluppare applicazioni e servizi dedicati. Vediamo gli aspetti principali di ognuna di queste:

- *Telefonia*

Pur dovendosi difendere dall'attacco sul mercato di nuovi concorrenti che prima fornivano altri tipi di servizio è anche vero che potrebbero essere il settore con maggiori vantaggi. La loro radicata presenza sul territorio e l'esperienza maturata con servizi di connettività dial-up e a banda larga, permette loro di sfruttare le nuove possibilità semplicemente aggiornando ciò che attualmente viene fornito. Il RG per sua natura supporta in se già le connessioni standard ed è predisposto per l'aggiornamento con moduli avanzati.

- *Tv via cavo, Tv digitale*

I cavi a disposizione di questa tecnologia che entrano all'interno delle abitazioni possono offrire molto di più dei semplici twisted pairs. Questo può essere un vantaggio diretto per questo tipo di industria che conta su tecnologie di cablaggio ormai consolidate ed affidabili. Quello che di nuovo potranno offrire spazia fra distribuzione di video in digitale, video-on-demand, internet a banda larga ed eventualmente servizio di telefonia.

- *Energia*

Purtroppo l'utilizzo dei RG da parte di questo tipo di industria prevede un drastico aggiornamento delle tecnologie utilizzate per il trasporto e per la connessione con le abitazioni. E' anche vero che negli ultimi anni sono stati fatti molti passi avanti e da un po' di tempo vengono sperimentate molte forme di comunicazione e fornitura distribuzione di servizi attraverso la rete elettrica sia all'interno di reti domestiche sia provenienti da reti esterne.

- *Computer*

Lo sviluppo di nuove tecnologie Hardware e Software sarà indispensabile per l'utilizzo di tutte le potenzialità e le capacità date dai RG. Inoltre si possono prevedere gli impieghi negli ambienti domestici di piccole reti di computer o di singoli computer col compito di gestione della rete.

- *Apparecchiature elettroniche ed elettrodomestici*

Anche questa industria beneficerà senza dubbio della presenza dei RG, e forse sarà quella più rivoluzionata. Potranno essere progettati e messi sul mercato nuovi tipi di dispositivi capaci di interagire con la rete domestica ed allo stesso tempo di dialogare con il mondo esterno usando in entrambe i casi il RG come interfaccia di scambio.

- *Sicurezza della casa*

Il RG avrà una notevole importanza nella home security, potrà essere facilmente introdotto un controllo della sicurezza sia delle cose che delle persone a distanza da postazioni remote fisse o mobili.

- *Controllo della casa*

Attraverso l'interconnessione della rete domestica al mondo esterno come per la sicurezza anche per questo tipo di industria si avranno benefici con innumerevoli nuovi servizi legati al controllo ed alla manutenzione degli impianti e degli ambienti che potranno svilupparsi con estrema semplicità.

- *Home networking*

Una delle caratteristiche chiave dei RG è quella di fornire intrinsecamente nel suo stesso concetto le basi per avviare attività legate all'home networking e all'home business grazie alla completa interazione dell'ambiente domestico con le reti esterne.

3. PRINCIPALI STANDARD

Esistono attualmente numerosi standard legati all'automazione e alla comunicazione domestica. Purtroppo questo è stato uno dei principali fattori di rallentamento per la crescita e per lo sviluppo di questo settore. Nell'ambiente domestico oggi sono presenti numerosissimi dispositivi, ognuno dei quali possiede protocolli di comunicazioni, portanti trasmissive, bit rate che li rendono molto spesso non compatibili ne tra loro ne con la rete esterna.

Su vari fronti si sta cercando di operare per eliminare questo problema, due delle iniziative a maggior peso nel settore della ricerca in questo senso sono quelle legate all'OSGi e all'UPnP.

3.1. OSGi

3.1.1. Caratteristiche

La *Open Services Gateway Iniziative* nacque nel Marzo del 1999 ad opera di un gruppo di oltre 60 compagnie tra le quali giganti nel settore della comunicazione come Sun Microsystems, Ericsson, Cisco, Nokia, Siemens, IBM, Motorola, Nortel, Philips, Oracle, Alcatel, Lucent, Toshiba, Texas Instruments, e altri. L'idea di base che accompagnò fin dall'inizio il progetto fu quella di standardizzare un Residential Gateway capace di connettersi con un vasto spettro di applicazioni per lo scambio di informazioni, in modo da fornire una interfaccia tra gli apparati domestici e le reti dati esistenti. La distribuzione delle specifiche fornite da questo gruppo avviene in modo libero ed indipendente. L'OSGi cerca di coinvolgere non solo le case produttrici ma anche l'utenza finale mediante la pubblicazione di articoli informativi e testi di riferimento o con attività educative. La ricerca si è concentrata fin da subito su una soluzione end-to-end fra i dispositivi della rete domestica ed i provider dei servizi.

Il componente centrale di questa struttura è il *service gateway* che funziona da piattaforma per tutti quei servizi basati sulla comunicazione. Il service gateway permette la gestione di audio, video, dati Internet e multimediali provenienti dalla rete domestica o ad essa destinati. Può inoltre funzionare come un application server per una vasta serie di servizi ad alto livello come la gestione dell'energia, della sicurezza, della manutenzione, commercio elettronico e quant'altro. Il service gateway rappresenta per i provider il punto focale di riferimento al quale mirare per fornire tutte quelle tipologie di servizi avanzati nuovi o gia esistenti.

La comunicazione tra provider ed abitazione è permessa grazie alla specifica di una *Application Program Interface* (API), messa liberamente a disposizione dei programmatori e scritta in linguaggio JAVA e che quindi può, almeno in teoria, essere esportata su qualunque tipo di piattaforma. Come si può ben capire le caratteristiche che rendono forte ed universale l'iniziativa dell'OSGi sono l'indipendenza da ogni tipo di supporto e la standardizzazione degli aspetti legati alla comunicazione.

3.1.2. Vantaggi legati all'utilizzo di OSGi

I benefici introdotti da OSGi sono innumerevoli e riconosciuti anche dalle più grandi case costruttrici di apparecchi domestici e dai fornitori di servizi. I vantaggi che rimangono più evidenti sono legati alla standardizzazione, ma non devono essere sottovalutati altri aspetti legati allo sviluppo organizzato ad API per questo tipo di Residential Gateway:

- *Indipendenza dalla piattaforma*
  OSGi può essere implementato su qualunque tipo di piattaforma, sia essa hardware o basata su sistema operativo, in questo modo si toglie al service gateway una eventuale dipendenza da un unico tipologia di mercato.

- *Indipendenza dall'applicazione*
  L'uso delle API permette al Residential Gateway di essere valido per un vasto spettro di applicazioni concepite anche in modo del tutto diverso. Con questo anche se esso era stato originariamente pensato per il mercato domestico, non è detto che il suo sviluppo non si possa tendere anche verso altri contesti.

- *Sicurezza*
  Le specifiche OSGi offrono svariati livelli di sicurezza del sistema permettendo così meccanismi basilari come la firma digitale dei servizi scaricati e il controllo sull'origine dei dati.

- *Servizi Multipli*
  Si permette a molti servizi anche sostanzialmente diversi tra loro di risiedere su un unico service gateway, che in questo modo diventa l'unico punto di riferimento all'interno dell'area domestica per tutti i fornitori di servizi.

- *Molteplicità di tecnologie legate alle reti locali*
  Le specifiche fornite dall'OSGi danno la possibilità di supportare con estrema semplicità una molteplicità sia di nuove che già consolidate tecnologie di comunicazione anche profondamente diverse tra loro per motivi legati ai cablaggi ed ai sistemi di trasmissione come BluTooth, HAVi, HomePNA, HomeRF, IEEE-1394, LonWorks, powerline communication system, USB, VESA, tecnologie wireless e molte altre.

- *Molteplicità di tecnologie legate alla progettazione di dispositivi*
  Riferendosi all'idea base di implementare gli aspetti legati ai servizi la specifica OSGi supporta svariati standard come UPnP e Jini che vanno a far parte direttamente delle API. Questo permette di utilizzare sulla rete dispositivi di concezione diversa tra loro anche a livello di progettazione e che usano protocolli interni differenti.

- *Coesistenza di svariati standard*
  Le specifiche OSGi sono state progettate propriamente per supportare il numero più elevato possibile di standard in circolazione siano essi utili alla comunicazione all'interno dell'area domestica siano essi per la connessione con le reti esterne ed i fornitori di servizi. Inoltre esiste la possibilità di espansione per eventuali nuove tecnologie mediante l'aggiunta di appositi moduli.

3.1.3. Descrizione dell'architettura di OSGi

Le specifiche tecniche OSGi definiscono uno standard per la progettazione di dispositivi domestici per l'utente finale con la possibilità di profonde personalizzazioni da parte dei costruttori. Queste permettono implementare funzionalità avanzate come scaricare software, gestire il ciclo di vita delle applicazioni, gestire la sicurezza del gateway, gestire l'accesso ai dispositivi connessi, amministrare delle risorse disponibili ed avere a disposizione strumenti per l'amministrazione remota del gateway.

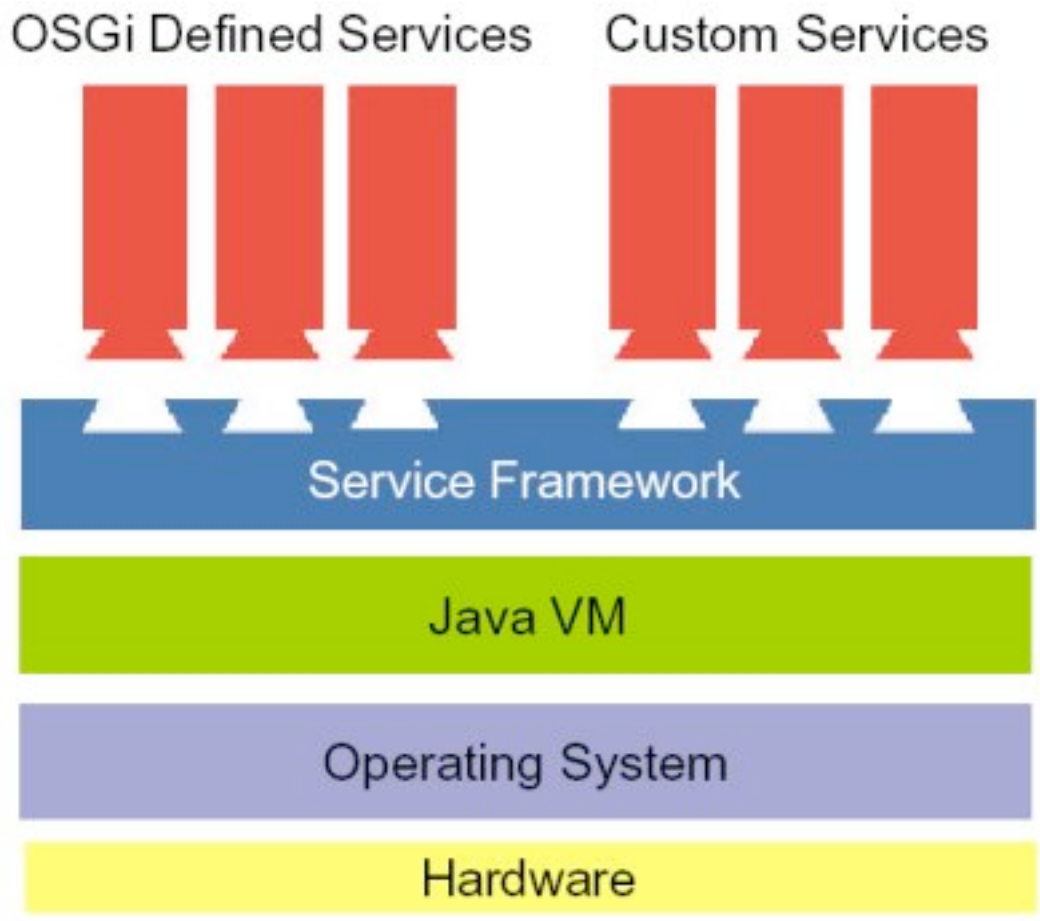


Figura 4: Localizzazione dell'architettura OSGi

Un'architettura OSGi del tipo end-to-end completa è composta principalmente da 6 componenti principali che sono: Services Gateway, Service Provider, Service Aggregator, Gateway Operator, WAN ed ISP, Reti locali e Dispositivi.

3.1.3.1. Services Gateway

Il componente centrale della struttura OSGi, gestisce le comunicazioni multimediali fra la rete domestica e le reti esterne e quelle interne alla rete stessa. Può allo stesso tempo funzionare da application server per un vasto campo di servizi ad alto livello come il controllo dell'energia elettrica, della sicurezza e molti altri. Tecnicamente non è altro che un server embedded che connette la rete geografica con la rete domestica e che fisicamente separa ed interfaccia le due diverse tipologie di rete.

Le specifiche OSGi per questo componente includono anche le API per la gestione del ciclo di vita delle applicazioni, per la gestione dei dati, della dipendenza fra i servizi, delle risorse e della sicurezza. Comunque sarà analizzato più nel dettaglio nei paragrafi successivi.

3.1.3.2. Service Provider

Nell'ambiente delle reti domestiche è la figura che fornisce dei servizi al consumatore. Questo è reso possibile tecnicamente attraverso il download di opportuni pacchetti software direttamente sul Residential Gateway. I livelli di sicurezza all'interno del modello OSGi facilitano il riconoscimento della provenienza dei dati scaricati e quindi la loro originalità. Una volta presenti sul RG i pacchetti gestiscono lo scambio di informazioni relative a quella tipologia di servizio fra i dispositivi interni e il fornitore esterno.

3.1.3.3. Service Aggregator

Le intenzioni del progetto OSGi sono state fin dall'inizio quelle di fornire le basi affinché nascano nuovi tipi di servizio avanzato per l'utente dove più servizi tradizionali abbiano la possibilità di interagire e collaborare tra di loro. Questo componente ha la responsabilità di gestire il rapporto fra i vari servizi e controllare che essi siano compatibili e non creino conflitti all'interno del Residential Gateway.

3.1.3.4. Gateway Operator

Molte volte associato al Service Aggregator questo componente ha le funzioni di gestione e manutenzione del Residential Gateway e dei suoi servizi. Tipicamente ha il compito di far partire, fermare, aggiornare e rimuovere componenti del Residential Gateway e controllare che tutto questo sia andato a buon fine e che non ci siano problemi. Inoltre ha l'onere di controllare sullo scambio di informazioni con i Service Provider e del download di nuovi servizi.

3.1.3.5. Reti geografiche e compagnie telefoniche/ISP

Questo rappresenta la piattaforma di comunicazione necessaria a far dialogare il Residential Gateway con l'esterno che è fornita dagli operatori di telefonia o dai fornitori di servizi di reti geografiche. Nei casi in cui l'esterno sia Internet questo componente è rappresentato dagli ISP (Internet Service Provider).

3.1.3.6. Reti locali e dispositivi

L'ultimo importante componente all'interno delle specifiche OSGi è rappresentato dalla rete locale domestica e da tutti i dispositivi ad essa connessi direttamente o indirettamente qualunque sia lo standard di comunicazione utilizzato e supportato.

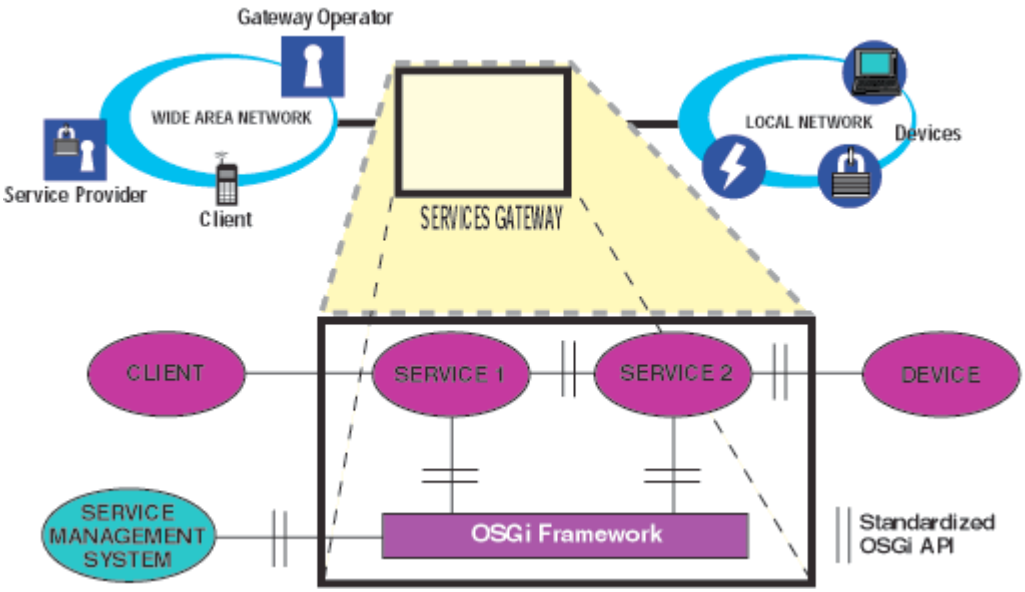


Figura 5: Specifiche e framework dell'architettura OSGi

3.1.4. Descrizione del Services Gateway

Uno degli aspetti sul quale il progetto OSGi ha concentrato maggiormente i propri sforzi è stato quello di implementare delle API all'interno del gateway e di renderle disponibili ai programmatori. Le API definite sono basate sul linguaggio Java in quanto ritenuto il più flessibile per sviluppare Services Gateway aperti e il più possibile esportabili ed ampliabili.

Un service gateway basato su Java deve necessariamente comprendere i seguenti aspetti:

- *Ambiente di sviluppo Java*

Per garantire un vasto spettro di mercati e di applicazioni possibili per i residential gateway l'uso di un ambiente Java per le specifiche delle API è la soluzione ideale. Per questi gateway le API sono conformi alla versione Java 2. L'obiettivo che si cerca di raggiungere all'interno del progetto è quello di rendere disponibile l'implementazione di Services Gateway sui Personal Java (pJava) e sulla versione Java 2 Micro Editino (J2ME) e su altri ambienti compatibili con il runtime di Java.

- *Infrastruttura di servizio*

Fornisce agli sviluppatori ed ai programmatori il contesto all'interno del quale scrivere il codice da eseguire sul gateway. All'interno di questa infrastruttura i servizi possono essere liberamente mandati in esecuzione e fermati, inoltre sono aggiornati dinamicamente, ed è concessa la possibilità di dialogare con gli altri servizi disponibili.

Gli aspetti positivi di questo ambiente sono la possibilità di caricare e aggiornare i servizi durante l'esecuzione, senza dover fermare l'ambiente, inoltre questo può essere usato anche su dispositivi con memoria limitata, offre un consistente numero di modelli di programmazione per gli sviluppatori dei servizi, gestisce le dipendenze tra i servizi ed è altamente scalabile.

Il miglior modo per sfruttare le potenzialità dell'ambiente è quello di progettare le applicazioni come un set di servizi ciascuno dei quali implementi una piccola parte del funzionamento globale. Questi poi saranno combinati insieme all'interno di un *Bundle* e caricati all'interno di un gateway. Un bundle non è altro che un Java Archive (JAR) che contiene le risorse, classi o tipi di dati, che implementano i servizi, tiene traccia dello stato delle interazioni fra le risorse coinvolte, può eventualmente contenere le classi per l'installazione, la configurazione, l'avvio e l'aggiornamento dell'infrastruttura, e infine dichiara quali classi devono essere usate per avviare o terminare un servizio.

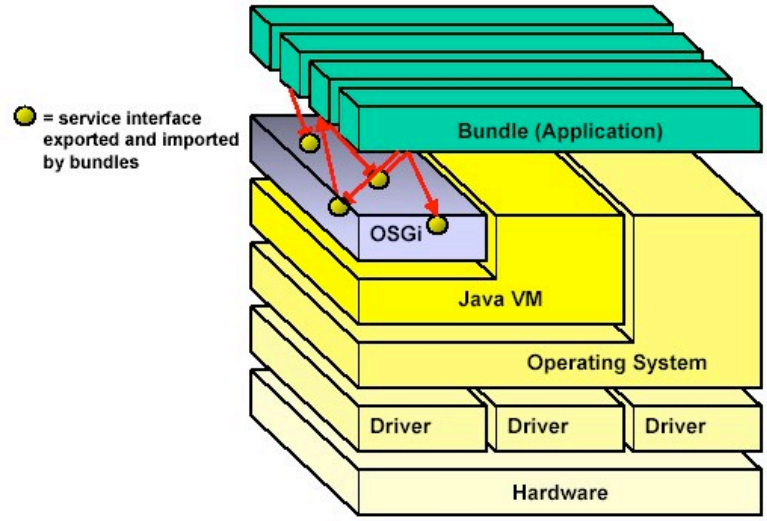


Figura 6: Ambiente OSGI

- *Gestore dell'accesso ai dispositivi*

Questo componente fornisce su richiesta e dinamicamente i servizi caricati sia al Services Gateway che all'hardware di rete. Il gestore dell'accesso ai dispositivi ha funzioni di supporto ai services provider che hanno il bisogno di accedere e di comunicare con i dispositivi e le applicazioni senza voler essere coinvolti nei dettagli di comunicazione a basso livello. Lavorando ad alto livello le sue principali caratteristiche sono l'indipendenza da specifiche architetture di rete, l'indipendenza da specifici dispositivi fisici, inoltre ha la possibilità di trovare automaticamente nuovi dispositivi connessi ed infine supporta le vecchie tipologie di rete esistenti gia prima delle specifiche OSGi che non potevano supportare il riconoscimento automatico.

Utilizza il concetto di *bundle di rete* che è un tipo speciale di bundle di infrastruttura di servizio che contiene gli stack dei protocolli, i driver e le altre risorse per la comunicazione. Inoltre utilizza il concetto di *bundle di dispositivo* che contiene il codice necessario per la comunicazione con tipi specifici di dispositivo.

- *Servizio di log*

Questo, che è l'unico servizio essenziale, definisce una API di servizio che permette agli altri servizi di scrivere e leggere dei file di log. Il servizio di log è di notevole importanza per gli sviluppatori per permettere loro la risoluzione dei problemi.

Oltre a questi servizi necessari, all'interno di tutti i Residential Gateway esiste anche una serie di servizi opzionali, anch'essi definiti all'interno delle specifiche:

- *Servizio HTTP*

Prevedendo che molti servizi in futuro saranno essenzialmente basati sui protocolli web è stata introdotta una API per il web server http, che è parte integrante del Services Gateway. Grazie a questa API diventa molto semplice per i programmatori configurare un server, pubblicare contenuti statici e dinamici generati da Java Serverlets.

Eccone le principali caratteristiche:

- Supporto per le specifiche API Java Serverlet 2.1 (o superiori);
- Un motore Java Serverlet per fornire l'ambiente a runtime per le Java Serverlet;
- Supporto per i protocolli HTTP/1.0 o HTTP/1.1 con la raccomandazione che la versione 1.0 supporti la funzionalità Keep-Alive della 1.1;
- Una API che permetta ai servizi di rimuovere o inserire dinamicamente testo statico, Java serverlets o altre risorse all'interno dello spazio URI dell'http server;
- Una API che permetta ai servizi con contenuto sensibile di disporre di una forma di autenticazione per verificare l'identità del richiedente e che dia la possibilità di creare connessioni sicure (crittografia SSL).

- *Servizio di accesso al client*

Occasionalmente i servizi in esecuzione sui gateway necessitano di interfacciarsi con l'utente finale, alcune interazioni possono essere gestite direttamente dal server remoto. In questi casi si ha il bisogno di permettere all'utente finale di poter accedere, leggere e modificare le informazioni sul gateway in modo uniforme e consistente.

Il servizio di accesso al client deve fornire un ambiente nel quale l'utente possa operare liberamente con interfacce standard e che tolga agli sviluppatori l'onere di gestire l'interazione. Supporta inoltre molti formati di dati e l'accesso mediante dispositivi mobili.

- *Servizio di configurazione dei dati*

Alcuni servizi necessitano di informazioni addizionali per la configurazione rispetto a quelle standard, queste possono esser più o meno strutturate e di diversa complessità. In aiuto agli sviluppatori il servizio di configurazione dei dati fornisce una API che standardizza l'accesso a questi dati e il loro inserimento.

- *Servizio per la persistenza dei dati*

Molti servizi hanno la necessità di poter utilizzare e scambiare con altri dei dati che perdurino nel tempo. Utilizzando questa API ogni servizio lo ritenga necessario può generare qualunque tipo di dato persistente, può cercare le informazioni richieste mediante richieste ad alto livello, può

correggere e cambiare dati e richieste, e può anche sincronizzare le proprie informazioni con un database posto sul server.

3.2. UPnP

3.2.1. Caratteristiche

Universal Plug and Play (UPnP) è un'architettura per la connettività pervasiva a reti peer-to-peer di apparecchiature intelligenti, dispositivi wireless, e PC di ogni genere. E' stata creata con lo scopo di portare facilità d'uso, flessibilità e basi standard nelle reti ad-hoc e domestiche, degli uffici, degli spazi pubblici, delle piccole industrie o connesse ad Internet che non abbiano un amministratore. E' un'architettura distribuita aperta che fa leva su tecnologie chiave del Web come TCP/IP ed altre per la gestione del trasferimento di dati fra apparecchiature di rete all'interno di reti domestiche.

UPnP è più di una semplice estensione del Plug and Play, è progettato per supportare reti a configurazione nulla (*zero-configuration)*, reti invisibili (*invisibile networking)*, e riconoscimento automatico di una vasta gamma di dispositivi. Questo significa che ogni apparecchiatura può entrare a far parte della rete, ottenere un indirizzo IP, trasmettere le sue referenze e ricevere informazioni sulla presenza e le capacità degli altri dispositivi presenti in rete il tutto dinamicamente senza il bisogno di un intervento esterno. Se la rete lo richiede sono supportati e possono essere utilizzati server DHCP o DNS. Inoltre ogni componente della rete può in qualunque momento disconnettersi senza per questo apportare problemi di configurazione di tutto l'apparato.

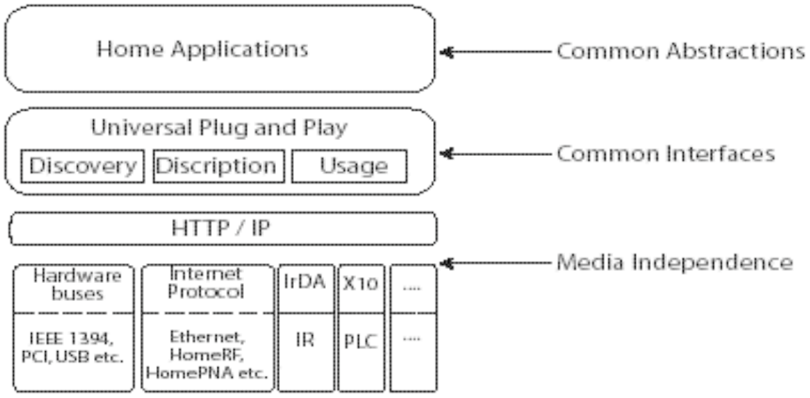


Figura 7: Architettura ad alto livello dell'UPnP

L'universalità di UPnP è data dalla mancanza di driver specifici per i dispositivi e per l'uso di protocolli comuni largamente diffusi. I dispositivi possono essere programmati mediante un qualsiasi linguaggio di programmazione e possono risiedere su un qualunque sistema operativo o sistema embedded.

Diversamente da quanto accadeva con OSGi la cui filosofia era quella di puntare ad un dispositivo centrale nella rete con funzione coordinatrice, UPnP punta a dotare ogni sistema dello strato software necessario per intraprendere le comunicazioni verso l'esterno o verso altre apparecchiature. Un'altra sostanziale differenza sta nel fatto che OSGi fornisce direttamente delle API agli stessi sviluppatori, la filosofia UPnP è invece quella di fornire delle linee guida e delle regole costruttive che gli sviluppatori sono tenuti a rispettare categoricamente. Dopo la fase di progettazione e costruzione ogni dispositivo deve passare dei rigidi test e soltanto dopo il loro superamento può ottenere la certificazione di compatibilità UPnP.

3.2.2. UPnP forum

Il forum UPnP è una iniziativa industriale nata con lo scopo di gestire facilmente e in maniera robusta la connessione fra i singoli dispositivi e i PC di qualunque sistema ed architettura essi siano dotati. Questa organizzazione cerca di creare degli standard che diano descrizioni precise dei

protocolli e degli schemi XML ai quali lo sviluppo dei dispositivi deve sottostare in modo da farli interoperare tra di loro in un ambiente di rete scalabile.

Con questo forum si cercano di dare delle linee guida ben precise da far rispettare alle singole case costruttrici e software houses in modo che ognuna di esse immetta sul mercato prodotti capaci di dialogare con gli altri senza la necessità di una infrastruttura che faccia da interfaccia di comunicazione.

UPnP fornisce agli sviluppatori una serie di interfacce per accedere e fornire servizi alle reti domestiche. Una delle sue caratteristiche chiave è rappresentata dalla indipendenza da strutture fisiche di rete con la funzione di mediatore. Supporta, inoltre, moltissimi standard per il trasporto delle informazioni trai quali l'Ethernet 10baseT e tutte le nuovissime tecnologie wireless.

3.2.3. Vantaggi legati all'uso di UPnP

I vantaggi introdotti da questo standard sono innumerevoli, i più importanti sono:

- *Standard Aperto*
  Ha protocolli relativamente semplici e standard simili a quelli definiti dalla Internet Engeneering Task Force (IETF). Il TCP/IP, su cui fa riferimento, permette la comunicazione tra moltissimi tipi di piattaforme esistenti e da la possibilità di essere compatibile con una vastità di dispositivi siano essi PC o elettrodomestici intelligenti.
- *Scalabilità*
  Normalmente è adatto per lavorare con reti piccole ma nulla vieta di poter utilizzare questo standard su reti molto più grandi.
- *Plug and Play*
  Molti utenti vogliono poter inserire nella rete un dispositivo ed utilizzarlo fin da subito senza bisogno di configurazioni o installazioni. UPnP è basato su dei protocolli che dinamicamente ed automaticamente gestiscono questo tipo di operazioni.
- Scarse risorse
  Le risorse richieste al sistema sono ridottissime, questo permette a UPnP di poter lavorare sia su PC che su dispositivi dotati di microcontrollori con scarse risorse hardware.
- *Ambienti multi piattaforma*
  Immaginando una casa con dispositivi capaci di interagire tra loro, lo standard UPnP è concepito sulla possibilità di interazione e scambio di informazioni fra varie tipologie di applicazione.
- *Integrazione con applicazioni gia esistenti e non basate su IP*
  La scelta di basarsi sul protocollo IP è una forte presa di posizione di UPnP che però non perde d'occhio tutte quelle applicazioni che non supportano questo protocollo web come le reti di intrattenimento basate su IEEE 1394.
- *Architettura non incentrata su PC*
  La configurazione di base di una rete UPnP può essere basata su un architettura di rete peer-to-peer, il che vuol dire che la rete domestica può lavorare senza un PC. Anche se questo non significa che in una rete la presenza di PC non abbia un ruolo importante.

3.2.4. Componenti di una rete UPnP

I componenti fondamentali per una rete UPnP sono tre e sono Periferiche, Servizi e Punti di Controllo.

3.2.4.1. Periferiche

Una periferica, per le specifiche UPnP, non è altro che un contenitore di servizi o di altre periferiche annidate. Ogni categoria di periferica avrà ad essa associato un particolare gruppo di servizi, ad esempio i servizi per le stampanti saranno in generale diversi da quelli per i TV color. I vari comitati facenti parte dell'UPnP Forum hanno il compito di specificare per ogni categoria i

rispettivi servizi e fornire poi dei documenti di descrizione in formato XML, che contengono anche informazioni addizionali come icone ed informazioni del costruttore.

3.2.4.2. Servizi

Questa entità mette a disposizione delle azione grazie alle quali modella le sue variabili interne. Anche queste hanno associato un documento XML di descrizione standard fornito dall'UPnP Forum che racchiude tra le altre informazioni l'url al documento di descrizione della periferica a cui è associato. Ogni periferica può avere più servizi.

Ogni servizio è caratterizzato da tre elementi: una tabella di stato, che ne definisce lo stato interno mediante l'aggiornamento costante delle variabili di stato interne; un server di controllo, che riceve azioni le esegue sulle variabili di stato e ne restituisce i risultati; un server degli eventi, che invia informazioni ai punti di controllo interessati ogni qual volta venga cambiato uno stato interno di cui sia utile saperne il cambiamento.

3.2.4.3. Punti di Controllo

E' un controllore capace di gestire e di rilevare le altre periferiche sulla rete. Dopo il rilevamento può: rilevare una descrizione della periferica e dei servizi ad essa associati, cercare dei servizi in base alle descrizioni pervenute, richiamare azioni per controllare i servizi, registrare un cambiamento di una variabile di stato di un qualche servizio.

In caso di realizzazioni di reti peer-to-peer le periferiche avranno al loro interno funzioni di punto di controllo, così come i punti di controllo avranno al loro interno funzioni di periferiche.

3.2.5. Supporti di rete e protocolli di comunicazione

La comunicazione nel protocollo UPnP è basata su IP standard, questo da la possibilità di utilizzare un qualunque tipo di tecnologia di rete esistente, come la linea telefonica, i cavi coassiali, la radiofrequenza, le linee elettriche, l'infrarosso, e di poter supportare in futuro anche nuove tecnologie purché la loro banda di comunicazione lo permetta. Altri tipi di dispositivi basati su standard trasmissivi del tipo HAVi, CeBus, LonWorks, EIB e X10 potranno essere interconnesse a queste reti semplicemente mediante l'uso di appositi bridge di semplice costituzione.

Le fondamenta dello standard UPnP si basano su un largo numero di standard industriali molto diffusi, tra cui:

- TCP/IP (Trasmission Control Protocol/Internet Protocol)
- DNS (Domain Name System)
- HTTP (HyperText Transfer Protocol)
- HTML (HyperText Markup Language)
- UDP (User Datagram Protocol)
- LDAP (Lightweight Directory Access Protocol)
- XML (eXtensible Markup Language)
- XSL (eXtensible Stylesheet Language)
- ARP (Address Resolution Protocol)

Altri importanti protocolli con importanza di rilievo sono HTTPU e HTTPMU che sono varianti del protocollo HTTP, definite per il recapito di messaggi via UDP/IP anziché TCP/IP.

Il protocollo SSDP (Simple Service Discovery Protocol) definisce le modalità di rilevazione dei servizi di una rete. Il protocollo SSDP si basa sui protocolli HTTPU e HTTPMU e definisce i metodi che consentono sia a un punto di controllo di individuare le risorse di interesse nella rete, sia alle periferiche di comunicare la loro disponibilità nella rete.

L'architettura GENA (Generic Event Notification Architecture) è stata realizzata per consentire l'invio e la ricezione di notifiche utilizzando il protocollo HTTP su TCP/IP e UDP multicast. L'architettura GENA definisce inoltre i concetti di sottoscrittori (subscriber) e autori (publisher) di notifiche per la generazione di eventi.

Il protocollo SOAP (Simple Object Access Protocol) definisce l'utilizzo degli standard XML (Extensible Markup Language) e HTTP per l'esecuzione di chiamate a procedure remote (RPC, Remote Procedure Call).

3.2.6. Fasi della connettività di una rete UPnP

Quando un qualunque dispositivo viene a collegarsi scatta un meccanismo che con passi successivi ne permette la configurazione con la rete e con gli altri dispositivi.

3.2.6.1. Indirizzamento

E' il passo iniziale del networking UPnP, durante questa fase ai dispositivi connessi viene assegnato un indirizzo con il quale saranno poi riconosciuti. Alla base delle reti UPnP c'è l'indirizzamento IP, ogni device deve disporre di un client DHCP e deve cercare un DHCP server appena dopo essersi connesso alla rete. In caso questo sia presente deve farsi assegnare un indirizzo IP altrimenti deve ricorrere all'auto assegnamento mediante il protocollo Auto IP. Per procedere con l'auto assegnamento i passi da compiere sono:

- Decidere se utilizzare l'Auto IP inviando sulla rete un messaggio "dcpdiscover". Se viene ricevuto un messaggio "dhcpoffer" entro un tempo determinato vuol dire che il servizio DHCP è disponibile altrimenti bisogna far ricorso all'Auto IP.
- Scegliere un indirizzo mediante un algoritmo predefinito nell'insieme degli indirizzi 169.254/16, tenendo di conto che gli ultimi e i primi 256 sono occupati.
- Testare l'indirizzo mediante un'interrogazione della rete che sfrutti il protocollo ARP. In caso l'indirizzo sia gia utilizzato se ne deve scegliere un'altro.
- Controllo periodico per valutare la disponibilità del DHCP server mediante messaggi "dchpdiscover", infatti, ogni dispositivo deve periodicamente controllare che non sia disponibile l'assegnazione dinamica degli indirizzi.
- Bisogna garantire il supporto per DNS e naming dei dispositivi in quanto per la comunicazione fra dispositivi è molto più efficiente sfruttare nomi simbolici più che indirizzi IP. Inoltre potrebbero esistere periferiche che hanno applicazioni ai livelli superiori all'UPnP che necessitano di nomi simbolici.
- Ogni dispositivo deve supportare le richieste di risoluzione Nome-IP poste da server DNS in accordo con quanto è previsto nelle RFC1034 e RFC1035.

3.2.6.2. Rilevazione

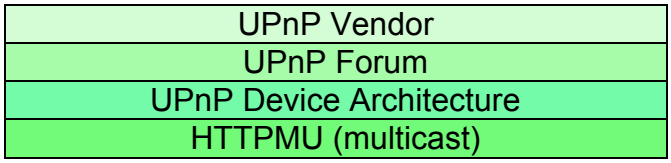
Durante questa fase il punto di controllo (*control point*) cerca dispositivi interessati alla connessione.

Quando un nuovo elemento si connette alla rete il protocollo prevede che esso invii un certo numero di messaggi in multicast sulla rete in modo da comunicare la sua disponibilità ad eventuali control point interessati. Allo stesso modo, anche un nuovo control point inserito su una rete deve interrogare la rete con messaggi multicast e rimanere in ascolto per eventuali risposte per vedere di quali servizi o dispositivi può disporre. L' unica aspetto che cambia in questi casi è il formato del messaggio e le informazioni in esso contenuto. Se un control point ha bisogno di informazioni ausiliarie sulle caratteristiche dei dispositivi, deve usare le informazioni contenute nel messaggio ricevuto per inviare una query di tipo *description* ed attendere la risposta.

Per evitare problemi di congestione il protocollo prevede un Time To Live (TTL) dei pacchetti IP impostato a 4 ma questo può essere riconfigurato. Gli indirizzi multicast standard, così come i meccanismi per la ricerca, l'interrogazione e la revoca sono tutti descritti dal Simple Service Discovery Protocol (SSDP).

I singoli passaggi della fase della rilevazione sono:

- *Annuncio(Advertising)*



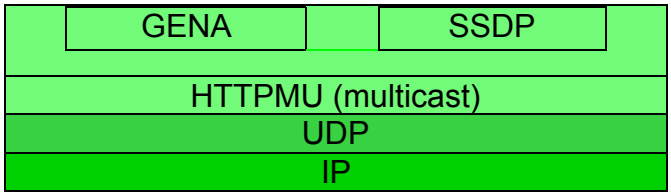


Figura 8: Livelli dello stack UPnP utilizzati da un messaggio di advertising della fase discovery

In questa fase è previsto l'invio di un messaggio che utilizza uno stack del protocollo ben definito. Nello strato più alto il protocollo prevede le informazioni specifiche del produttore come l'identificatore o la descrizione del dispositivo. Scendendo nello stack si trovano alcune informazioni fornite dal comitato di lavoro UPnP Forum come ad esempio il tipo di dispositivo. Lo strato successivo è definito nel protocollo UPnP stresso. I due strati successivi sono dei varianti dei messaggi multicast HTTP estesi usando i metodi del General Event Notification Architetture (GENA), degli header del Simple Service Discovery Protocol (SSDP). Questi messaggi sono poi inoltrati per via UDP mediante l'uso di IP.

- *Ricerca*

Lo stack di utilizzato nella ricerca è molto simile al precedente, con l'unica differenza che il quarto e quinto livello non utilizzano il GENA ma solo SSDP.

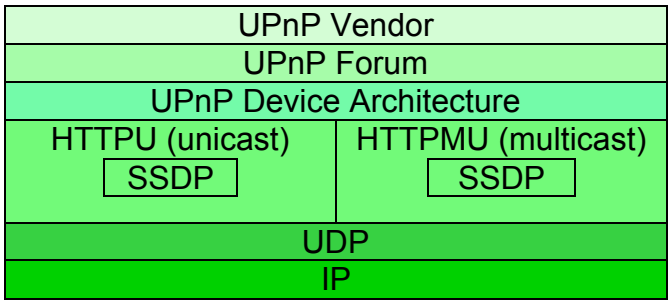


Figura 9: Livelli dello stack UPnP utilizzati da un messaggio di tipo search della fase discovery

3.2.6.3. Descrizione

Quando un control point viene a conoscenza della presenza di un device ne conosce pochissime informazioni, ossia solo quelle presenti nei messaggi di risposta tra le quali anche l'indirizzo della descrizione UPnP. Tramite questo indirizzo si possono raggiungere nuove e più approfondite informazioni.

La descrizione è suddivisa in due parti logiche: la *device description* che è un contenitore delle informazioni fisiche e logiche, ed una o più *service description* che include le capacità espresse dal dispositivo. La device description contiene le specifiche del costruttore, alcune informazioni di progettazione come il tipo di modello, numeri seriali, indirizzo web del costruttore, ecc. Per ogni servizio fornito dal dispositivo la device description elenca il nome e li tipo del servizio, l'indirizzo della service description, un indirizzo per il controllo ed uno per gli eventi. Inoltre include tutte le informazioni sui dispositivi embedded e l'indirizzo della loro descrizione.

La device description è fornita dal costruttore ed espressa in sintassi XML.

Una service description ha una lista di comandi, detti anche azioni, al quale il servizio risponde, e dei parametri, o anche argomenti, per ogni azione. Inoltre include una serie di variabili descritte con tipo, range di valori ed eventi che le modificano, utili per il funzionamento del servizio a runtime.

Anche questi sono scritti in XML dalle case costruttrici e si basano su dei template forniti dal comitato UPnP Forum.

```
<deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
<friendlyName>short user-friendly title</friendlyName>
<manufacturer>manufacturer name</manufacturer>
<manufacturerURL>URL to manufacturer site</manufacturerURL>
<modelDescription>long user-friendly title</modelDescription>
<modelName>model name</modelName>
<modelNumber>model number</modelNumber>
<modelURL>URL to model site</modelURL>
<serialNumber>manufacturer's serial number</serialNumber>
<UDN>uuid:UUID</UDN>
<UPC>Universal Product Code</UPC>
<iconList>
  <icon>
```

Figura 10: esempio di parte di codice XML di una device description

Per trovare informazioni ulteriori sulle potenzialità e le funzionalità di un dispositivo l'operazione da fare è una semplice richiesta di tipo HTTP GET all'indirizzo fornito dal dispositivo nel discovery message.

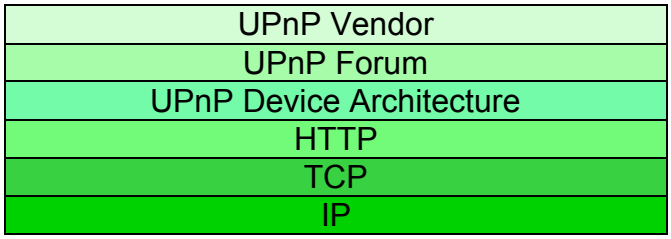


Figura 11: Livelli dello stack UPnP utilizzati da un messaggio di tipo description

3.2.6.4. Controllo

Durante questa fase il control point invoca azioni sui dispositivi o sui suoi servizi con una sorta di chiamata di procedure remote e ne registra i valori ritornati, sia che essi siano risultati validi o fallimenti, così da approfondire le proprie conoscenze.

Per invocare un azione su un servizio di un dispositivo il control point deve inviare un messaggio di tipo *control* all'indirizzo di controllo del servizio. In conseguenza a questa azione il servizio ritornerà un codice di errore o un risultato. L'effetto dell'azione può essere modificato andando ad agire sulle variabili che descrivono lo stato di runtime. Per registrare lo stato delle variabili il control point deve esplicitamente richiederlo con delle query appropriate ed acquisirne il valore di ritorno. Se dopo un lasso di tempo prestabilito non si ricevono risposte si può assumere che il servizio non è più disponibile.

```
POST path of control URL HTTP/1.1
HOST: host of control URL:port of control URL
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:serviceType:v#actionName"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:actionName xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
     <argumentName>in arg value</argumentName>
     other in args and their values go here, if any
   </u:actionName>
```

```
  </s:Body>
</s:Envelope>
```

Figura 12: Esempio di richiesta per una azione

```
HTTP/1.1 200 OK
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: when response was generated
EXT:
SERVER: OS/version UPnP/1.0 product/version

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:actionNameResponse xmlns:u="urn:schemas-upnp-org:service:serviceType:v">
     <argumentName>out arg value</argumentName>
     other out args and their values go here, if any
   </u:actionNameResponse>
 </s:Body>
</s:Envelope>
```

Figura 13: Esempio di possibile risposta da parte di un servizio

I messaggi di questo tipo usano degli strati diversi dello stack del protocollo UPnP. Diversamente da quanto avveniva per gli advertising stavolta i messaggi sono distribuiti via HTTP mediante TCP sfruttando l'IP. Inoltre i messaggi sono formattati mediante gli standard Simple Object Access Protocol (SOAP).

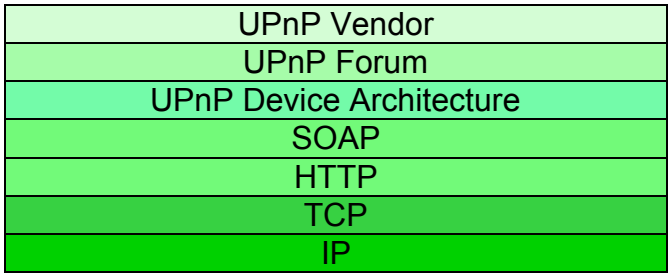


Figura 14: Livelli dello stack UPnP utilizzati da un messaggio di tipo control

3.2.6.5. Gestione degli eventi

In questa fase il control point sta attento ad eventuali cambiamenti di stato all'interno dei dispositivi.

L'essenziale per un control point è trovare un dispositivo e ricavarne la descrizione insieme a quella dei suoi servizi, e questo è fatto durante le fasi di descrizione e di controllo.

Se una o più variabili di quelle presenti nella service description cambiano il loro stato ne viene comunicato il cambiamento dallo stesso servizio. In questa sessione chiamando *publisher* il sorgente dell'evento (tipicamente un servizio di device) e *subscriber* la destinazione dell'evento (tipicamente un control point). Per iniziare un colloqui il subsriber deve inviare un *subscription message*, ad esso il publisher risponderà fornendo una parametro che è la durata del dialogo. Per

mantenere attiva la conversazione il subscriver dovrà rinnovare questa volontà prima che l'intervallo di tempo finisca, qualora l'intenzione sia quella di abbandonare basterà far terminare il tempo senza inviare alcun rinnovo.

Per rendere noto il cambiamento delle variabili il publisher deve inviare degli *event messages* che possono contenere una o più variabili ed il loro valore attuale, il tutto espresso con sintassi XML. Un messaggio speciale è quello inviato all'inizio della fase, nel quale sono contenute tutte le variabili ed i loro stati iniziali, che permette al subscriber di inizializzare il suo modello dello stato del servizio. Per supportare la presenza di più control point contemporaneamente i publisher possono inviare messaggi non più contenenti solo le variabili modificate, ossia un sottoinsieme, bensì tutte le variabili con il loro stato, saranno poi i singoli control point a selezionare le informazioni a loro utili. Alcune variabili possono cambiare il loro valore troppo rapidamente o contenere valori troppo ingombranti, per questo si applica una specie di filtro che fa in modo che esse non siano inviate in ogni messaggio ma ad intervalli di tempo regolari, o che la loro lettura avvenga solo su richiesta esplicita del control point.

L'invio e la ricezione di messaggi di tipo subscription e event usano il sottoinsieme di livelli dello stack del protocollo UPnP descritto in figura.

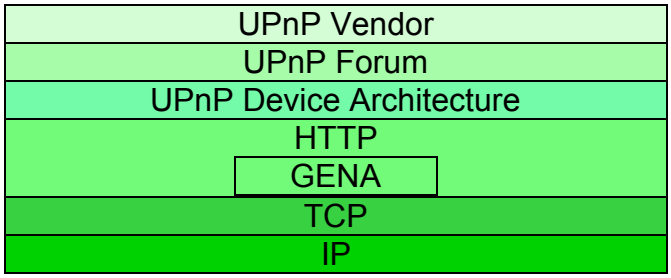


Figura 15: Livelli dello stack UPnP utilizzati da un messaggio di tipo event e subscription

Questi messaggi sono distribuiti via HTTP esteso coi metodi e gli header della General Event Notification Architecture (GENA), mediante TCP sfruttando l'IP.

Vediamo adesso più nel dettaglio due aspetti:

Subscription

Se un servizio supporta la gestione degli eventi esso pubblica degli event message per i subscriber interessati. Il publisher mantiene una lista con tutti i subscriber e per ciascuno di essi mantiene le seguenti informazioni:

- *Identificatore unico del subscriber* che deve essere unico universalmente in modo da garantire l'assenza di ambiguità.
- *Indirizzo di consegna per gli event message* che deve essere unico ed è necessario per far giungere i messaggi a destinazione.
- *Chiave dell'evento* è un identificatore che è inizializzato a zero si incrementa per ogni event message della sequenza.
- *Durata del collegamento* che rappresenta il periodo di tempo della durata del collegamento.

```
SUBSCRIBE publisher path HTTP/1.1
HOST: publisher host:publisher port
CALLBACK: <delivery URL>
NT: upnp:event
TIMEOUT: Second-requested subscription duration
---
HTTP/1.1 200 OK
DATE: when response was generated
SERVER: OS/version UPnP/1.0 product/version
SID: uuid:subscription-UUID
```

```
TIMEOUT: Second-actual subscription duration
```

Figura 16: Esempio di una richiesta di tipo subscription ed una possibile risposta

Ogni qualvolta un subscriber invia una richiesta e questa è accettata il publisher risponde con il suo identificatore unico e la durata. Subito dopo il publisher invierà il primo event message comunicando il nome ed il valore corrente di tutte le sue variabili. Per mantenere attiva la connessione, prima della fine dell'intervallo di tempo dovrà inviare un messaggio di tipo *renewal* al solito indirizzo a cui inviava quelli di tipo event.

Quando una connessione termina l'identificatore non è più valido e ad un messaggio di tipo event si avrà una risposta di errore. La terminazione prima del tempo può essere ottenuta mediante un messaggio di tipo *cancellation*.

Event Message

Questo tipo di messaggi include i cambiamenti avvenuti su una o più variabili. Devono essere inviati il più celermente possibile rispetto all'evento e possono comunicare singoli o multipli cambiamenti con lo stesso messaggio.

```
NOTIFY delivery path HTTP/1.1
HOST: delivery host:delivery port
CONTENT-TYPE: text/xml
CONTENT-LENGTH: Bytes in body
NT: upnp:event
NTS: upnp:propchange
SID: uuid:subscription-UUID
SEQ: event key

<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
  <e:property>
    <variableName>new value</variableName>
  </e:property>
  Other variable names and values (if any) go here.
</e:propertyset>
```

Figura 17: Esempio di una richiesta di tipo event che usa il metodo NOTIFY

Ognuno di questi messaggi è identificato da una chiave, questo permette un più facile gestione riconoscimento degli errori nella sequenza. La chiave è un intero implementato su 4 byte (32 bit) e al momento in cui il contatore raggiunge il suo massimo riparte da 1 e non da 0.

Se non si ricevono risposte dal subscriber agli event message, il publisher può decidere di continuare ad inviare messaggi fino alla fine del tempo o provare a disconnettersi e riconnettersi, perdendo però l'identificativo iniziale.

Entrambe i tipi di messaggio sono standardizzati dall'UPnP Forum mediante degli appositi template che prevedono strutture fisse ben precise ed una serie di argomenti predefiniti. Questi schemi sono scritti con sintassi XML e prevedono due parti, la prima dedicata alle strutture e la seconda ai tipi di dato.

3.2.6.6. Presentazione

Questa è l'ultima fase del networking UPnP e fornisce una semplice interfaccia utente basata su HTML per controllare e/o modificare lo stato di un dispositivo.

Se il dispositivo possiede un indirizzo per la presentazione (presentation URL) il control point può recuperare una pagina da questo indirizzo sulla quale basarsi. Il grado di completezza delle informazioni contenute su queste pagine dipende direttamente dai singoli dispositivi e dalla complessità della loro presentation page. L'indirizzo della pagina è contenuto all'interno della device description che è fornito mediante una messaggio di tipo description.

Raggiungere la pagina necessita di una semplice richiesta HTTP che usa un sottoinsieme dello stack del protocollo UPnP che possiamo vedere in figura.

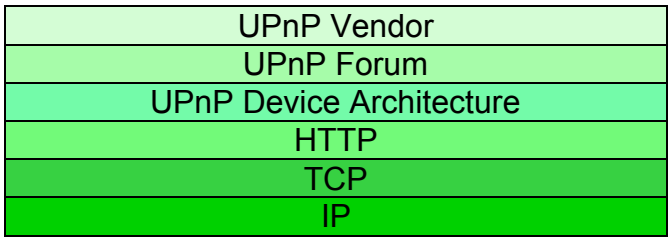


Figura 18: Livelli dello stack UPnP utilizzati per recuperare una pagina di presentazione.

La pagina di presentazione non è standardizzata dal Forum UPnP e quindi è gestita interamente dal costruttore del dispositivo o dal fornitore del servizio. Essa deve essere scritta in HTML 3.0 o versioni più recenti e non ha limiti ne sull'impaginazione ne sui contenuti.

4. INTEL SOFTWARE FOR UPnP TECHNOLOGY

"*Reduce your development costs and speed time to market! Download Intel software for UPnP technology, including authoring and remote I/O tools for UPnP devices.*" Con questo proclama si apre la home page di presentazione del software che viene distribuito liberamente dalla Intel per gli sviluppatori di device UPnP. Poche parole che però descrivono molto bene lo scopo che i vari research and development team della Intel si sono dati da quando, nel 1999, hanno iniziato a lavorare al progetto UPnP per una connettività digitale per l'utenza domestica che sia facilmente utilizzabile. Una prima release del suo *development kit for UPnP technology* è stata distribuita nel 2000 sottoforma di open source.

Intel è stata una delle società fondatrici del UPnP Forum e ad oggi ha una partecipazione attiva nello studio e nella realizzazione di nuove soluzioni software capaci di fornire agli sviluppatori solide basi di partenza per la creazione di device e control point basati sullo stack UPnP. Allo sviluppatore sono forniti liberamente tre pacchetti:

- *Intel Authoring Tools for UPnP Technologies*

Questo kit di strumenti include *Intel Device Builder* un tool ideato appositamente per generare velocemente device e control point che siano allo stesso tempo personalizzabili, efficienti, portabili e facili da usare. Le piattaforme di sviluppo supportate sono molteplici e fra queste le principali sono Linux, Microsoft Windows e Microsoft Pocket PC. Sono distribuiti anche dei codici sorgente di esempio di varie applicazioni.

- *Intel Remote I/O for UPnP Tecnologies*

Questo insieme di strumenti fa vedere allo sviluppatore come è resa possibile la comunicazione attraverso la rete e fra i dispositivi. E' infatti capace di tracciare e simulare delle comunicazioni interne ad una rete UPnP ed analizzarne gli esiti mediante report e statistiche.

- *Intel Tools for UPnP Tecnhologies*

Basato sul framework di Microsoft .NET questo kit di strumenti aiuta in fase di progettazioni in compiti come lo sviluppo, il test e la conoscenza dei dispositivi così da velocizzare il lavoro. Importante è il pieno supporto per le distribuzioni media AV.

4.1. Intel Authoring Tools for UPnP Technologies

I problemi che più da vicino riguardano lo sviluppatore al momento della progettazione e della creazione di dispositivi basati sullo stack UPnP iniziano prima di tutto dalla comprensione della tecnologia UPnP. Infatti, questo framework è relativamente nuovo e la perdita di tempo legata all'apprendimento costa tempo e denaro. Il tool messo appunto dalla Intel toglie al programmatore questo onere fornendo un'interfaccia intuitiva che riduce al minimo il bagaglio di conoscenze da avere.

Un altro problema è dovuto all'interoperabilità tra i vari device che solitamente non è garantita neppure tenendo sotto strette osservanze le linee guida di UPnP. Questo software standardizza e rende sempre possibile l'interoperabilità che deve essere un requisito fondamentale da rispettare.

Gli ultimi due aspetti che riguardano da vicino gli sviluppatori sono legati alle dimensioni del codice e del programma a run-time, e all'ottimizzazione delle performance di esecuzione, che devono essere in grado di garantire l'esecuzione e la portabilità su ogni tipo di piattaforma hardware e software qualunque siano le risorse a disposizione.

L'implementazione di tecnologie UPnP all'interno di dispositivi di rete o di applicazioni software prevede tre passaggi base:

1_ La progettazione dell'interfaccia del control point o del dispositivo;
2_ L'implementazione di uno stack UPnP che soddisfi questa interfaccia;
3_ La validazione dello stack per la compilazione e l'interazione.

Questi passaggio sono tutti supportati e resi possibili dai tool della Intel che rappresentano una soluzione completa che permette agli sviluppatori di costruire dispositivi e control point altamente efficienti per molte piattaforme.

La soluzione presentata sta il questo software che non è altro che un *code wizard* che permette la costruzione in automatico del codice sorgente di un dispositivo lavorando mediante un'interfaccia grafica sui parametri che contraddistinguono l'applicazione. Questo componete include:

- *Intel Device Builder*

Può generare a partire da un set di descrizioni XML di servizi lo stack in linguaggio C di control point e device completamente funzionanti e personalizzabili. Semplicemente agendo su una serie di decisioni al momento della generazione del codice si ottiene uno codice finale efficiente e molto compatto.

Il ruolo dell'Intel Device Builder è quello di aggregare automaticamente dei Service Control Protocol Document (SCPDs) che siano ben strutturati e da questi generare il codice ebmbedded delle applicazioni. Questo permette al programmatore di incentrare tutti i propri sforzi sulla progettazione logica della struttura dei dispositivi senza dover pensare a quella dello stack UPnP.

Viene fornita la possibilità di creare sia semplici devices che control point, inoltre viene supportata la progettazione di device annidati in altri.

Il risultato finale è un listato di codice nel quale sono presenti commenti che permettano una facile personalizzazione. Tutto ciò associato ad una generazione automatica dei documenti XML, alla robustezza del codice generato ed alla mancanza di errori riesce a ridurre drasticamente il time-to-market dei prodotti e tolglie notevoli responsabilità ai singoli sviluppatori.

- *AV Microstacks*

Questo è un insieme di stack AV generati dal Device Builder, i quali rappresentano un buon punto di partenza per gli sviluppatori per iniziare a creare le proprie applicazioni. Le specifiche sono racchiuse e catalogate all'interno di una serie di documenti che contengono anche tutta una gamma di informazioni e dati di aiuto per la ricerca e per renderne più facile l'utilizzo.

Uno dei punti chiave degli AV Microstack è la riduzione della grandezza del codice e la dimensione ridotta a run-time. Per rendere evidente questo aspetto vengono forniti una serie di esempi dimostrativi disponibili ai programmatori da studiare ed analizzare.

- *Sample Applications*

Include una serie di esempi di applicazioni scritte in C/C++ per varie piattaforme in modo da dare delle basi di riferimento agli sviluppatori. Sono presenti dei codici sorgente per dimostrare come gli sviluppatori possano utilizzare gli stack UPnP e gli AV Microstack in una soluzione completa.

4.2. Intel Remote I/O for UPnP Tecnologies

Intel fornisce nel pacchetto di software questo toolkit basato sul framework di Microsoft .NET che dimostra in che modo questa tecnologia permette ad un computer, tramite un'interfaccia utente in remoto, di interagire con un altro dispositivo della rete. Include diversi esempi di applicazioni e di codice sorgente di servizi. All'interno del Remote I/O sono disponibili le seguenti componenti:

- *Remote I/O Server*

Questa applicazione scova sulla rete la presenza di dispositivi RemoteIO Client e tenta di risalirne alla appropriata interfaccia utente. Può anche essere utilizzato per controllare con quale interfaccia e da quale computer host un client ha iniziato a farsi riconoscere.

- *Remote I/O Clients*

Questi sono inclusi solo per le piattaforme Microsoft Windows e Microsoft PocketPC. E' uno stack per IO remoto, scritto in codice C, compreso per facilitare l'indirizzamento delle piattaforme embedded.

- *Remote I/O Control*

Questa applicazione cataloga tutti i RemoteIO Client di una rete e tutti i canali virtuali presenti per ognuno di essi. L'utente può, infatti, cambiare canale, disconnettere un client e inserire degli input da remoto.

- *Sample User Interfaces*

Fornisce, mediante del codice sorgente in C#, degli esempi di interfaccia utente per la televisione (640x480) a dispositivi palmari (240x320). Rappresentano esempi di come uno sviluppatore può progettare esperienze direttamente per l'utente finale.

4.3. Intel Tools for UPnP Tecnhologies

Basato sul Framework di Microsoft .NET, questi tool aiutano gli sviluppatori software ed i progettisti hardware a velocizzare i loro tempi di realizzazione, di testing e di apprendimento dei dispositivi più complessi, con particolare attenzione ai dispositivi audio-video. In questo pacchetto sono compresi:

- *Device Spy*

E' il primario UPnP Universal Control Point progettato con una interfaccia utente robusta, intuitiva e semplice da usare. Permette il riconoscimento dei device UPnP, e ne visualizza le informazioni e le azioni nel dettaglio, esegue il monitor degli eventi e gestisce gli errori.

- *Service Author*

E' un semplice strumento di authoring per documenti SCPD in XML. Permette la creazione e la manutenzione dei servizi in modo estremamente semplice. Crea il codice e ne controlla l'esattezza e la funzionalità e include un generatore di codice C#.

- *Device Sniffer*

E' lo strumento capace di monitorare tutto il traffico broadcast della rete, inoltre semplifica la risoluzione dei problemi legati all'interazione di tipo SSDP e può filtrare ed inviare pacchetti in questo formato.

- *Device Validator*

E' un tool pensato per il testing e può essere applicato a qualunque dispositivo UPnP e di servizio.

- *Device Scriptor*

E' un'applicazione di scripting, molto semplice da usare, che permette agli sviluppatori di ideare script di scenario e di eseguirli verso i dispositivi UPnP.

- *AV Media Controller*

E' un UPnP AV Control Point completo in ogni suo aspetto, è equivalente al Device Spy per gli AV. Ha la funzione di controllare tutto il contenuto delle directory della rete e dei vari formati delle informazioni contenute, visualizza tutto il contenuto multimediale in modo da favorire una rapida ricerca e la costruzione dinamica di play list per i programmi che supportano certe funzionalità. E' una delle applicazioni basilari del pacchetto.

- *AV Media Server*

E' un AV UPnP Content Directory completo in ogni funzionalità. Usa il filesystem come una base di dati e permette agli utenti di fare il drag & drop delle cartelle e del loro contenuto per rendere più facile lo scambio di materiale multimediale. La sua interfaccia tiene anche traccia dei download effettuati.

- *AV Media Render*

E' un UPnP AV renderer completo che supporta l'esecuzione in più istanze in contemporanea, le play list, la moderazione degli eventi, la ricerca delle tracce aduio, la ricerca della posizione all'interno di una traccia e molto altro.

- *AV Wizard*

E' un UPnP AV Control Point che permette l'esecuzione dei contenuti immagazzinati sul file system locale mediante dei riproduttori disponibili. Il punto di forza è che questa applicazione è integrata in Windows XP come una estensione della shell.

- *Network Light*

Anch'esso incluso nel pacchetto è una semplice applicazione a forma di lampadina che a comando si accende o si spenge che può essere utilizzata per dimostrare le potenzialità basilari di Device Spy e Service Author.

- *Device Relay*

Replica i dispositivi UPnP di una rete all'interno un'altra. Semplicemente eseguendo questa applicazione su due diverse sottoreti connesse da una rete WAN, in automatico verranno replicati i dispositivi di una nell'altra. I comandi e gli eventi sono replicati al dispositivo giusto mediante l'uso di pacchetti che sfruttano il protocollo SOAP.

- *Micro Tools for UPnP Technologies*

Il pacchetto chiamato "Intel Authoring Tools for UPnP Technologies" include dei codici sorgente completi di molte applicazioni, queste stesse applicazioni sono fornite sottoforma di file binari.

- *Tools for UPnP Technologies Java Edition*

Le tre applicazioni più usate "Device Spy", "Device Sniffer" e " Network Light" disponibili in una versione Java costruiti in JDK 1.3.

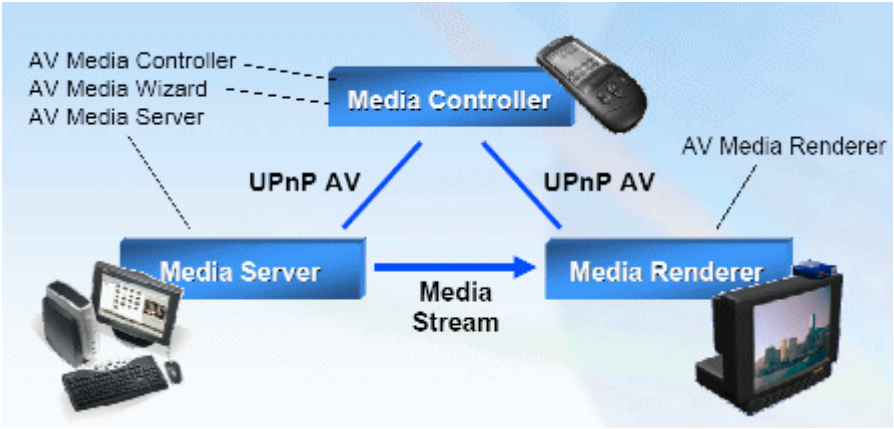


Figura 19: Architettura di un UPnP AV

4.3.1. Intel Micro Tools for UPnP Technologies

Questi tools sono inclusi sottoforma di codice sorgente nel pacchetto "Intel Authoring Tools for UPnP Technologies". Il codice è interamente generato con lo strumento di generazione "Intel Device Builder". Rappresentano piccoli esempi di applicazione che possono essere progettati e costruiti con il supporto di questi strumenti. All'interno del pacchetto "Intel tool for UPnP Technologies" sono forniti anche in formato binario. Sono tutte applicazioni compilate per Pocket PC basati su ARM anche se sono fornite versioni simili compilate per Windows.

Vediamoli nel dettaglio:

- *Device Scanner*

Permette agli utenti di visualizzare sul proprio PocketPC tutti i dispositivi presenti in rete con tutti i loro servizi e dispositivi embedded. E' simile al Device Spy ma non consente l'invocazione di azioni e la richiesta di eventi.

- *Micro Light*

Serve a vedere se un device risponde ad una interrogazione, semplicemente fa cambiare di colore il display in modo da testarne la funzionalità.

- *Micro Media Server*

E' un server multimediale che supporta le funzioni minime indispensabili come la condivisione della root di un dispositivo senza alcun tracciamento o altro.

- *Micro Media Browser*

Permette ai dispositivi PocketPC di trovare contenuto multimediale all'interno della rete partendo con la ricerca dalla root del proprio file system e procedendo sugli altri. Consente la sola navigazione del contenuto ma non l'utilizzo.

- *Micro Media Render*

Sui PocketPC utilizza il Media Player Control che deve essere presente separatamente e permette l'esecuzione di contenuto audio e video.

- *Micro Remote I/O Client*

Supporta il protocollo di visualizzazione remota XRT2 e rende disponibile ad un server remoto l'intero display di un dispositivo.

- *Micro Remote I/O Control*

Permette di listare tutti i Clients I/O remoti su una rete e per ognuno definirne le interfacce disponibili e scegliere quale di esse utilizzare.

- *Micro Remote I/O Server*

Oltre a listare tutti i Clients I/O remoti presenti su una rete rende loro disponibili alcune applicazioni di esempio.

5. COSTRUZIONE DI UN DEVICE

Il componente centrale degli Intel *Authoring Tools for UPnP Technology* è il *Device Builder* (DB). Il tool permette agli sviluppatori, con sole conoscenze basilari della tecnologia UPnP, di creare e gestire senza problemi sia device che control point che supportino stack completi ed adeguati. I passi da compiere per la progettazione e la realizzazione di un device o di un control point sono semplici e di facile realizzazione.

5.1. Progettazione dei servizi

Prima di poter usare il DB per generare lo stack di un dispositivo è necessario creare dei documenti SCPD scritti in linguaggio XML che definiscano i servizi da includervi. L'operazione di costruzione dello schema è resa automatica mediante un software presente nel pacchetto dei tool che si chiama *Intel Service Author*.

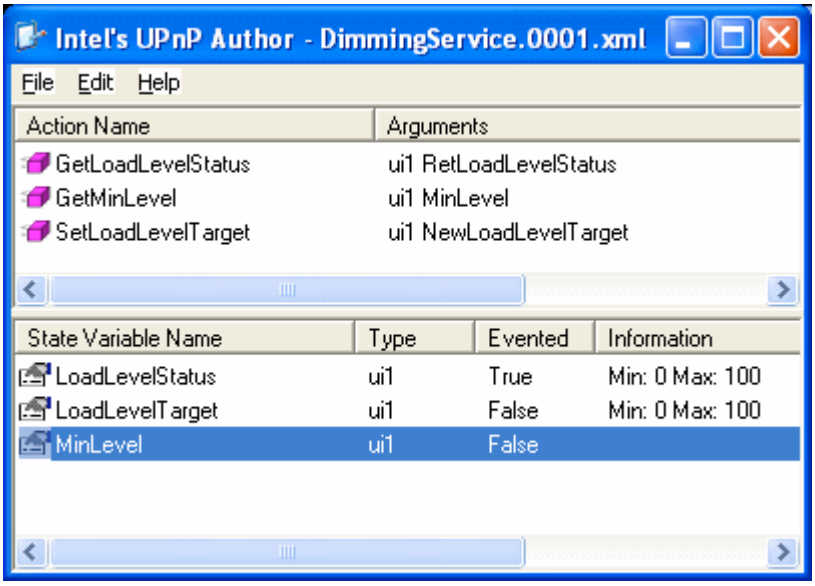


Figura 20: Interfaccia del tool Intel Service Author

La semplicità di utilizzo di questo tool permette di automatizzare l'immissione di Azioni e di Variabili di Stato mediante delle form predefinite e standard. Allo sviluppatore è permesso sia importare schemi preesistenti da modificare, che crearne di nuovi. L'inserimento di nuove variabili o azioni si ottiene con il semplice click, mediante il tasto destro del mouse, sull'area di lavoro corrispondente al nuovo elemento da creare.

Al termine dell'operazione il salvataggio avrà come output direttamente lo schema completo in XML della descrizione del dispositivo.

5.2. Generazione dello stack UPnP del dispositivo

Intel Device Builder permette allo sviluppatore di inserire descrizioni complete, in XML sottoforma di Service Control Point Document (SCPD), di servizi e di inserire device o Control Point embedded incluse nel dispositivo di partenza. Un dispositivo UPnP ha un device radice (root) dal quale possono dipendere dei servizi o direttamente altri dispositivi embedded che a loro volta diventano radice.

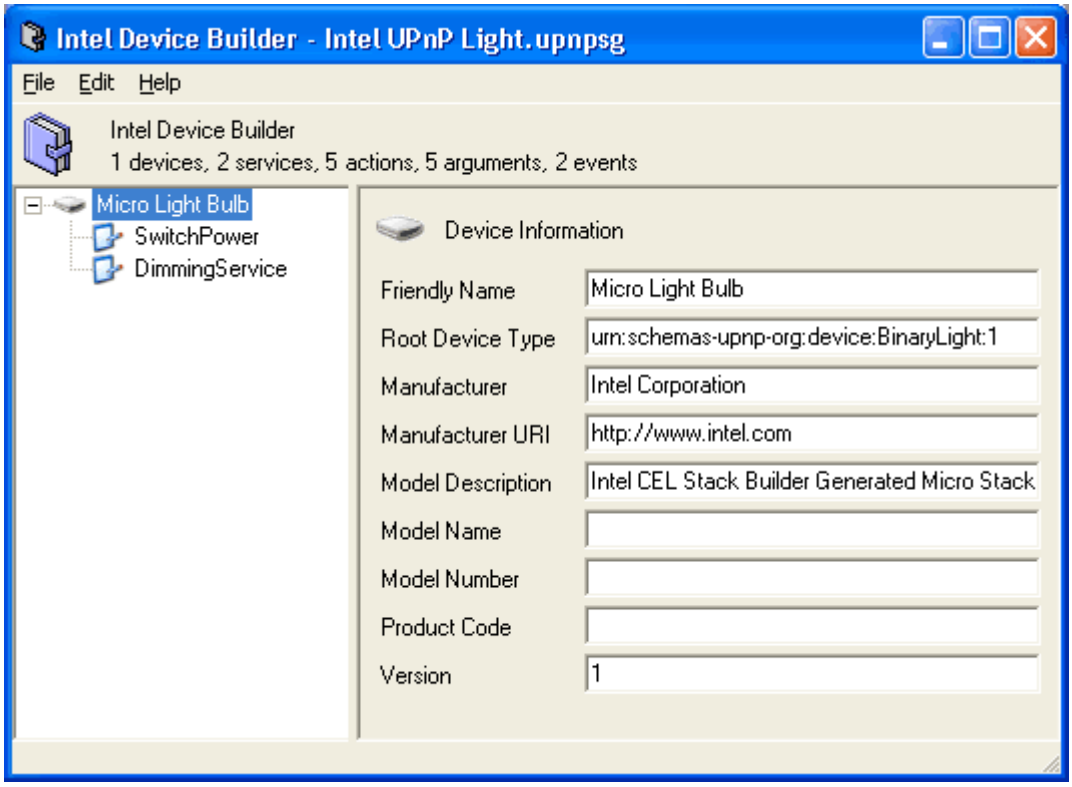


Figura 21: Interfaccia del Intel Device Builder. Vediamo sulla sinistra lo schema con il device root con alcuni servizi e la sua descrizone sulla destra.

L'aggiunta, la rimozione e la possibilità di importare nuovi servizi o device è supportata da operazioni automatizzate richiamabili mediante il menu in alto. Ogni singolo componente dello schema ha una descrizione interna raggiungibile e personalizzabile dall'interfaccia stessa attraverso la semplice selezione del componente. Uno dei campi più importanti ed essenziali è il Device Type che permette ai Control Point di selezionare e riconoscere le specifiche funzioni di un dispositivo.

Per quanto riguarda i servizi, oltre ad elencare le caratteristiche e l'elenco di azioni e variabili di stato, PB supporta l'utilizzo di nomi simbolici che facilitano il lavoro dello sviluppatore ma non sono direttamente richiesti dal protocollo UPnP.

Con questo tool i device possono essere salvati e riutilizzati in un secondo momento per apportare modifiche o essere utilizzati all'interno di altri device. Questo strumento da anche la possibilità di importare schemi preesistenti anche dalla rete.

In alcuni casi, un'azione può avere risposte molto pesanti in termini di quantità di dati. Generalmente per grandi quantità di dati si tende ad immagazzinare prima la risposta in memoria e poi ad inviarla sulla rete, si può però ottimizzare questa soluzione iniziando ad inviare il contenuto prima del termine mediante l'utilizzo di protocolli SOAP in grado di gestire la risposta frammentata. L'unico problema è che questa soluzione è incompatibile con la gestione della sicurezza da parte di UPnP e quindi deve essere utilizzata solo in casi specifici per operazioni non sensibili.

Una volta terminata la progettazione si può esportare lo stack del dispositivo o del control point mediante la voce corrispondente del menu. Questa operazione apre il *Device Generation Form* o il *Control Point Generation Form.*

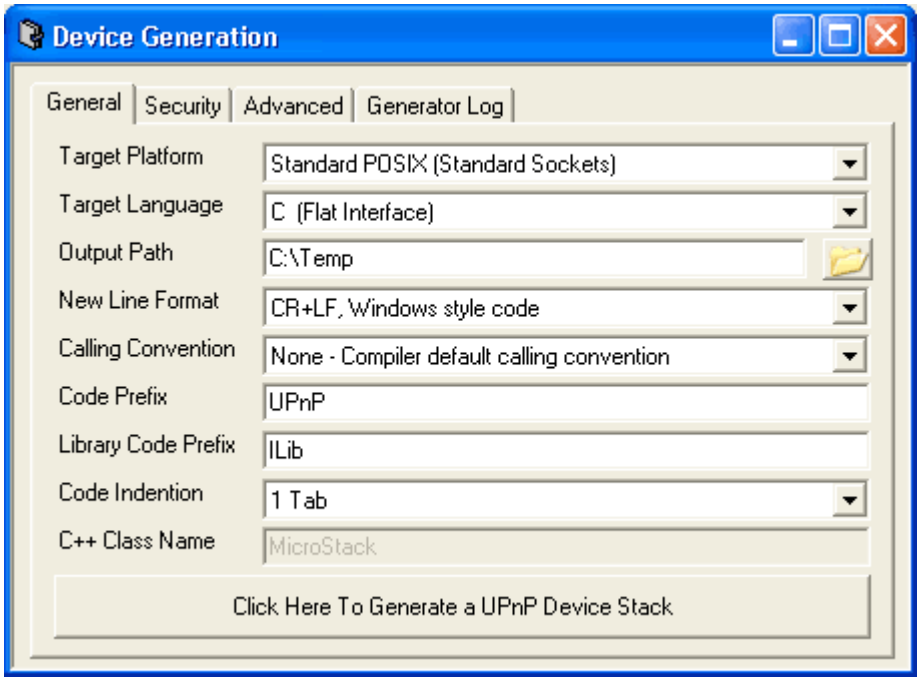


Figura 22: Interfaccia del Device Generation Form.

Nel pannello *General* sono contenute tutti principali settaggi:

- *Targhet Platform*: Da la possibilità di scegliere la piattaforma per la quale si vuol costruire il dispositivo. Quelle supportate sono:
  - *Microsoft Windows*: teoricamente il codice generato dovrebbe essere compatibile con tutte le versioni di Windows a partire dal 95 in poi e con entrambe le versioni di Winsock1 e Winsock2. L'esportazione per questa piattaforma genera del codice C/C++ compatibile con il Microsoft Visual Studio 7.
  - *PocketPc*: Il codice è stato testato sul PocketPC 2002 basato su ARM usando come ambiente di sviluppo il Microsoft Embedded Visual Studio 3. Il codice generato è di dimensioni molto ridotte ed ottimizzato in modo da fare un uso moderato delle risorse disponibili, per questo non prevede nessun processo in esecuzione in background.
  - *Posix*: Il codice è stato testato su sistemi POSIX Linux e Cygwin. Uno dei punti di forza degli stack generati per questo tipo di piattaforma è che è stato possibile utilizzarli anche su altre piattaforme non basate su POSIX.
  - *Microsoft .NET Framework*: Può essere generato anche del codice al top di un generico stack .NET.
- *Targhet Language*: Attualmente è supportato solo il C.
- *New Line Format*: Linux e POSIX utilizzano un formato CR, Windows e .NET invece utilizzano un formato CR+LF.
- *Calling Convention*: Serve per decidere il metodo standard di gestione delle chiamate.
- *Code Prefix*: Con questa stringa messa come prefisso verranno riconosciuti tutti i membri globali dello stack ed i nomi dei file dei moduli.
- *Library Code Prefix*: Questo prefisso distinguerà tutti i moduli ed i membri globali all'interno delle librerie.
- *Code Indentation*: E' la marcatura utilizzata per generare il codice.
- *C++ Class Name*: Attualmente questa opzione è inutilizzata in quanto non è previsto il supporto del C++.

Il pannello *Security* è utilizzato per gestire ed includere nel progetto alcune librerie di sicurezza standard ma opzionali che permettono la gestione dei permessi di accesso ad una risorsa.

Mediante il pannello *Advanced* si possono personalizzare alcuni aspetti strettamente legati al protocollo di trasmissione. Tra l'altro possono essere impostati i parametri riguardanti le dimensioni degli header e dei corpi dei pacchetti trasmessi, la gestione degli errori e delle temporizzazioni di trasmissione e la gestione delle chiamate dirette a funzioni disponibili su device esterni.

Il pannello *Generation Log* semplicemente tiene traccia dei passaggi avvenuti per la pubblicazione in modo da facilitare eventuali operazioni di debug.

5.3. Principali caratteristiche

Con questo sistema possono essere progettati dispositivi di qualunque tipo basati sul nucleo UPnP. Altri stack disponibili per gli sviluppatori sono molto più ingombranti e generici di quello fornito dalla Intel che ha impiegato tutte le sue forze e le sue potenzialità per la creazione di un ambiente specifico ed ottimizzato. La specializzazione permette maggiori prestazioni e minori complicanze sia in fase di sviluppo che in fase di utilizzo. Inoltre, il fatto che ogni applicazione generata sia eseguita in un unico processo fa si che il codice generato sia esportabile su quasi qualunque tipo di piattaforma.

L'ambiente garantisce anche la completa interoperabilità fra il dispositivo creato e gli altri basati su UPnP tanto da permettere la certificazione di ogni applicazione costruita basandosi sullo stack generato.

Le dimensioni tipiche di un device variano da circa 80k in ambienti Windows ai 70k per Posix e circa 60k per PocketPC, l'utilizzo della memoria è circa di 2-4 volte quello delle dimensioni del codice generato. Queste ottimizzazioni sono rese possibili dall'uso massiccio delle Device Descripon che permettano in fase di generazione di prendere decisioni che altrimenti andrebbero prese a runtime. Inoltre vengono utilizzate tecniche di modularizzazione e di standardizzazione come nel caso del protocollo HTTP che prevede degli header pre-generati.

In presenza di esecuzioni simultanee di più Microstack le dimensioni rimangono sempre ridotte in quanto le librerie utilizzate sono comuni a tutti i tipi di dispositivo e di control point e quindi sono caricate una volta sola.

Ogni processo in esecuzione è capace di compiere più operazioni simultaneamente, come ad esempio gestire 5 richieste in uscita e 5 in entrate del protocollo HTTP sia di versione 1.0 che di versione 1.1.

La gestione della sicurezza attualmente è implementata soltanto all'interno dei dispositivi e non nei control point. Per questo sono presenti alcune librerie personalizzabili attraverso dei parametri che permettono, ad un'azione che implementi la gestione della sicurezza, di essere direttamente gestite dalle librerie specializzate.

5.4. Microstack

Microstack è il nome dato ai moduli generati in prima istanza mediante Intel Device Builder per qualunque tipo di piattaforma. Al momento della generazione il Device Builder genera un'applicazione di esempio completa che include degli statement printf, una risposta di esempio e un esempio di errore.

```
void UPnPMathService_Add(void* upnptoken,int a,int b)
{
	printf("Invoke: UPnPMathService_Add(%d,%d);\r\n",a,b);

	/* TODO: Place Action Code Here... */

	/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
	UPnPResponse_MathService_Add(upnptoken,250);
}
```

Figura 23: Esempio di alcune righe di codice auto-generato dal Device Builder

Non ci sono garanzie che il codice generato sia del tutto valido, ad esempio il valore 250 dell'ultima riga è generico, deve essere poi personalizzato in base alle esigenze dello sviluppatore, che può apportare delle modifiche.

```
void UPnPMathService_Add(void* upnptoken,int a,int b)
{
        if ((a+b) < a)
        {
                UPnPResponse_Error(upnptoken,901,"Sum of A and B overflow int
                type");
        }
        else
        {
                UPnPResponse_MathService_Add(upnptoken,a+b);
        }
}
```

Figura 24: Esempio di personalizzazione degli errori HTTP.

Il programmatore deve sempre fornire un response ad una richiesta, in quanto questo è necessario per un corretto funzionamento delle comunicazioni all'interno dello stack. Le richieste non possono e non devono mai essere ignorate.

In molti casi, pur essendo il codice dei Microstack in linguaggio C, può sorgere l'esigenza di includere degli header C++.

```
extern "C"
{
   #include "UPnPMicroStack.h"
}
```

Figura 25: Esempio di inclusione di un header C++.

In molti casi pratici i device non vanno in esecuzione da soli, bensì sono accompagnati da uno o più control point. Per ridurre le dimensioni totali del codice a run-time si cerca il più possibile di tenere in comune a più esecuzioni lo stesso codice, come le librerie, e di usare per tutti lo stesso thread. In questo modo non si introduce overhead che altrimenti si ha in presenza di più thread a causa dei cambiamenti di stato ed al passaggio da un thread all'altro. Questa tecnica di programmazione, usata in tutti i microstck, prende il nome di "thread chaining" e si basa sul concetto che tutti i thread presenti devono avviarsi e fermarsi nello stesso momento.

Lo schema è molto semplice, c'è un singolo blocco di wait nel quale si aspetta per uno o più eventi che possono accadere. Quando uno di questi eventi viene invocato si torna indietro ad eseguire lo statement selezionato e si ritorna di nuovo in attesa di un nuovo evento. Ad ogni loop sono eseguiti piccoli task per la lettura dei dati, per la ricerca o con altre funzionalità.

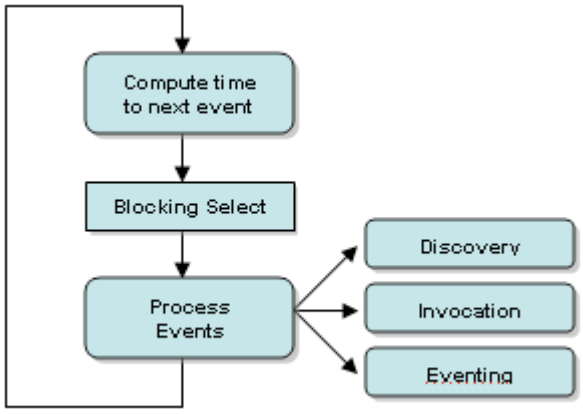


Figura 26: Schema di esecuzione di un thread

6. ANALISI DI UN ESEMPIO PRATICO

La progettazione e costruzione di device o servizi semplici o complessi è resa semplice grazie alla presenza, nel pacchetto fornito dalla Intel, di tutto il software necessario alla realizzazione passo passo di qualunque tipologia di dispositivo. Il passaggio dalla teoria alla pratica per lo sviluppatore appare quindi molto veloce e di facile comprensione, anche senza approfondite conoscenze del protocollo UPnP.

Analizziamo adesso le fasi della progettazione di un dispositivo reale chiamato *CD Player* la cui funzione è quella di riprodurre CD musicali. L'esempio segue le linee guida fornite direttamente dall'UPnP Forum.

Dei quattro servizi che fanno parte del progetto finale del Device viene descritta la fase di implementazione di uno solo di essi.

6.1. Progettazione di un servizio: Audio

Audio è il primo dei servizi di cui è composto CD Player ed è il modulo responsabile della gestione dei parametri della riproduzione audio.

6.1.1. Tipologia e descrizione del servizio

Audio è un servizio compatibile con la UPnP Device Architecture V.1.0 e rappresenta un semplice controllore di flusso audio. Permette di avere un controllo per il volume, il tono e il bilanciamento spaziale (sinistra-destra) di un dispositivo con un output di tipo audio.

Funzioni abilitate:

- Settaggio del volume
- Settaggio dei bassi
- Settaggio degli alti
- Settaggio del bilanciamento verso destra
- Settaggio del bilanciamento verso sinistra
- Settaggio del bilanciamento frontale

Non tratteremo aspetti più complessi come l'equalizzazione visto che non comporta differenze ai fini della progettazione concettuale.

Supporremo anche che il segnale audio analogico sia inviato su una qualunque porta fornita dal dispositivo e che eventuali azioni di switching tra più porte siano gestite da altri servizi esterni.

6.1.2. Definizione del modello del servizio

La definizione del modello del servizio Audio è resa possibile mediante l'utilizzo del tool *Service Author* che permette la creazione automatica della descrizione XML a partire dalla definizione guidata delle variabili di stato e delle azioni.

6.1.2.1. Service Type

L'identificatore del tipo di servizio che utilizzeremo è:

urn:schemas-upnp-org:service:Audio:1

Dove "Audio:1" è usato per l'appunto per la definizione di questo specifico tipo di servizio.

6.1.2.2. Variabili di stato

Le variabili di stato presenti in questo servizio sono:

- *Volume*
  Tipo di dato: Unsigned Integer 8 (ui1) – Valori abilitati: >=0, <=255 – Valore di default: 30
  Definisce la variabile di stato per il settaggio del volume di un riproduttore audio. Il valore è di default è impostato a 30 e non può scendere al di sotto di 0 (silenzio) o salire al di sopra di 255 (saturazione).
- *Treble*
  Tipo di dato: Integer 8 (i1) – Valori abilitati: >=-127, <=127 – Valore di default: 0
  Indica l'intensità del controllo del tono degli alti che può essere maggiore di zero quando si vuole enfatizzare questo parametro rispetto al tono medio o minore di zero quando lo si vuole diminuire. Il valore di default impostato a zero indica che il tono inizialmente non è modificato.
- *Bass*
  Tipo di dato: Integer 8 (i1) – Valori abilitati: >=-127, <=127 – Valore di default: 0
  Indica l'enfasi dei bassi rispetto al tono medio, il suo comportamento è identico a quello degli alti.
- *Balance*
  Tipo di dato: Integer 8 (i1) – Valori abilitati: >=-127, <=127 – Valore di default: 0
  Indica il bilanciamento spaziale verso l'uscita audio destra o verso quella sinistra.
- *Fade*
  Tipo di dato: Integer 8 (i1) – Valori abilitati: >=-127, <=127 – Valore di default: 0
  Indica il bilanciamento spaziale in senso frontale.
- *A_ARG_TYPE_Delta*
  Tipo di dato: Integer 16 (i2) – Valori abilitati: >=-255, <=255 – Valore di default: nessuno
  Definito solo per funzionare da argomento per alcune azioni che saranno definite successivamente. Non modella nessuno stato del servizio.

Ciascuna di queste variabili non ha relazioni con nessuna altra, sono tutte indipendenti tra loro.

6.1.2.3. Azioni

Le azioni di controllo sulle variabili di stato sono gestite in modo che in caso di errore non ci sia alcun cambiamento nello stato interno. Queste sono:

- *AddVolume*
  Incrementa e decrementa il valore numerico della variabile di stato Volume. Prende come argomento un valore numerico e ritorna il nuovo valore dello stato.
  Argomenti:

Delta	IN	A_ARG_TYPE_Delta	
NewVolume	OUT	Volume	(Valore di ritorno)

  Se Delta è positivo incrementa altrimenti decrementa il volume, se si raggiunge il massimo o il minimo non lo si oltrepassa e non si genera errore.
  Errori:

402	Invalid Argument
501	Action Failed

  La funzione è del tipo:

  ```
  IF (Volume+Delta > maximum) THEN ASSIGN(Volume,maximum)
  ELSE IF (Volume+Delta < minimum) THEN ASSIGN(Volume,minimum)
  ELSE ASSIGN(Volume,Volume+Delta)
  ```

- *SetVolume, SetTreble, SetBass, SetBalance, SetFade*
  Semplicemente assegnano alla variabile di stato corrispondente il valore impostato come argomento.
  Argomenti:

NewVolume		IN	Volume
NewTreble	IN		Treble
NewBass		IN	Bass
NewBalance		IN	Balance
NewFade		IN	Fade

  Errori:

402	Invalid Argument
501	Action Failed

  La funzione è del tipo:

  IF (InArg > maximum) THEN ASSIGN(StateVar, maximum)
  ELSE IF (InArg < minimum) THEN ASSIGN (StateVar, minimum)
  ELSE ASSIGN (StateVar, InArg)

- *GetAudio*
  Fa una richiesta per una delle variabili di stato del servizio.
  Argomenti:

CurrentVolume		OUT	Volume
CurrentTreble		OUT	Treble
CurrentBass		OUT	Bass
CurrentBalance		OUT	Balance
CurrentFade	OUT		Fade

  Errori:

501	Action Failed

  Possono esserci relazioni fra le varie azioni che possono eventualmente chiamarsi a vicenda.

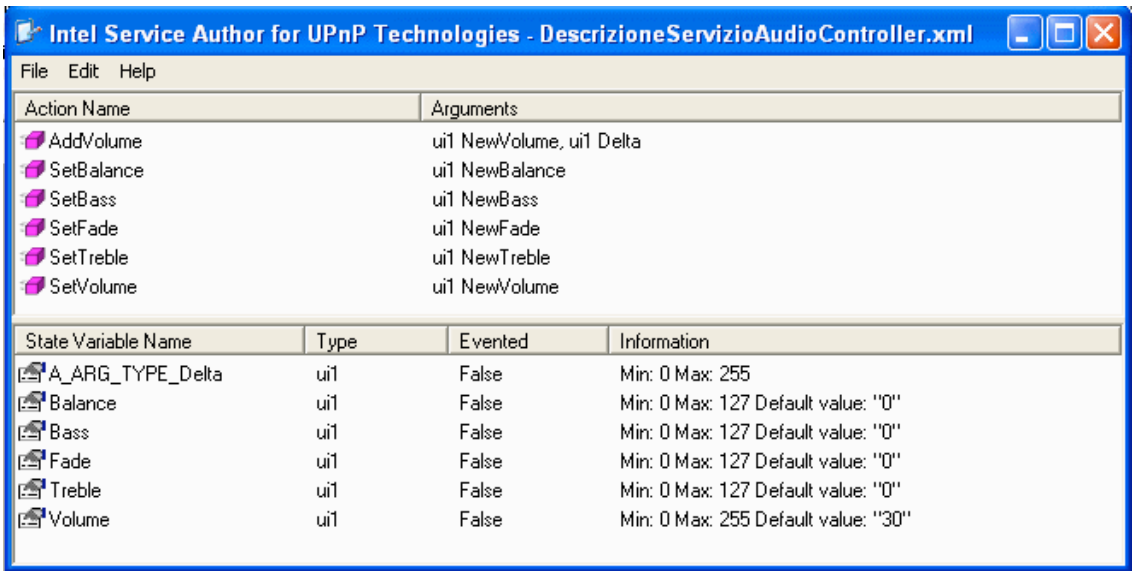


Figura 27: Schermata della finestra del tool Intel Service Author dopo la definizione delle Varibili di Stato e delle Azioni del servizio.

6.1.2.4. Descrizione XML del servizio

Settando le variabili di stato e le azioni che sono state elencate nei paragrafi precedenti utilizzando il tool "Intel Service Author for UPnP Technologies" viene generato in automatico un XML contenente la descrizione del servizio.

Il codice generato è il seguente:

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service:Audio:1">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>SetVolume</name>
      <argumentList>
        <argument>
          <name>NewVolume</name>
          <direction>in</direction>
          <relatedStateVariable>Volume</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>AddVolume</name>
      <argumentList>
        <argument>
          <name>NewVolume</name>
          <direction>in</direction>
          <relatedStateVariable>Volume</relatedStateVariable>
        </argument>
        <argument>
          <name>Delta</name>
          <direction>out</direction>
          <retval />
          <relatedStateVariable>A_ARG_TYPE_Delta</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetBalance</name>
      <argumentList>
        <argument>
          <name>NewBalance</name>
          <direction>in</direction>
          <relatedStateVariable>Balance</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetTreble</name>
      <argumentList>
        <argument>
          <name>NewTreble</name>
          <direction>in</direction>
          <relatedStateVariable>Treble</relatedStateVariable>
        </argument>
      </argumentList>
```

```
    </action>
    <action>
      <name>SetBass</name>
      <argumentList>
        <argument>
          <name>NewBass</name>
          <direction>in</direction>
          <relatedStateVariable>Bass</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetFade</name>
      <argumentList>
        <argument>
          <name>NewFade</name>
          <direction>in</direction>
          <relatedStateVariable>Fade</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>Volume</name>
      <dataType>ui1</dataType>
      <defaultValue>30</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>255</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>Bass</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>-127</minimum>
        <maximum>127</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>Fade</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>-127</minimum>
        <maximum>127</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>Balance</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>-127</minimum>
        <maximum>127</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>Treble</name>
      <dataType>ui1</dataType>
```

```
        <defaultValue>0</defaultValue>
        <allowedValueRange>
          <minimum>-127</minimum>
          <maximum>127</maximum>
        </allowedValueRange>
      </stateVariable>
      <stateVariable sendEvents="no">
        <name>A_ARG_TYPE_Delta</name>
        <dataType>ui1</dataType>
        <allowedValueRange>
          <minimum>-255</minimum>
          <maximum>255</maximum>
        </allowedValueRange>
      </stateVariable>
    </serviceStateTable>
</scpd>
```

6.1.2.5. Funzionamento teorico

Per permettere all'utente la gestione delle variabili di stato del dispositivo, un control point dovrà invocare delle azioni appropriate che agiscono sulla variabile interessata.

```
Invoke AddVolume with 10
```

Figura 28: Codice di esempio per aumentare il volume di 10.

```
Invoke SetVolume with 0
```

Figura 29: Codice di esempio per eliminare l'audio.

```
Invoke GetAudio
```

Figura 30: Esempio di richiesta per avere il valore della variabile di stato Volume

```
Use Current Treble out argument
```

Figura 31: Codice di esempio per inizializzare l'interfaccia utente.

6.2. Progettazione di un dispositivo: CD Player

La costruzione del dispositivo a partire dai file XML precostituiti dei servizi componenti è affidata al software "Intel Device Builder" che fa parte del pacchetto "*Intel Authoring Tools for UPnP Technologies*".

6.2.1. Tipologia e descrizione del dispositivo

Questo esempio rispetta le definizioni lo standard Universal Plug and Play V.1.0. CD Player è un lettore di Compact Disc (CD) che supporta tutte le funzioni basilari come il controllo del volume, del bilanciamento, dei toni dell'audio e invia il segnale all'esterno tramite delle connessioni analogiche.

6.2.2. Definizione del dispositivo

6.2.2.1. Device Type

L'identificatore usato per questo dispositivo è:

urn:schemas-upnp-org:service:CDPlayer:1

Dove "CDPlayer:1" è l'identificativo del dispositivo.

6.2.2.2. Modello del dispositivo

Il dispositivo ha la struttura visibile in tabella:

Device Type	Root
CD Player	Si

Service Type	Service ID	Impl.
SwitchPower:1	*SwitchPower*	No
ChangeDisc:1	*ChangeDisc*	No
PlayCD:1	*PlayCD*	No
Audio:1	*Audio*	Yes

Quando SwitchPower è in stato OFF non sono gestite le richieste provenienti dall'esterno e tutti gli altri servizi sono inattivi. L'inattività dei servizi non genera errori di risposta ad eventuali richieste.

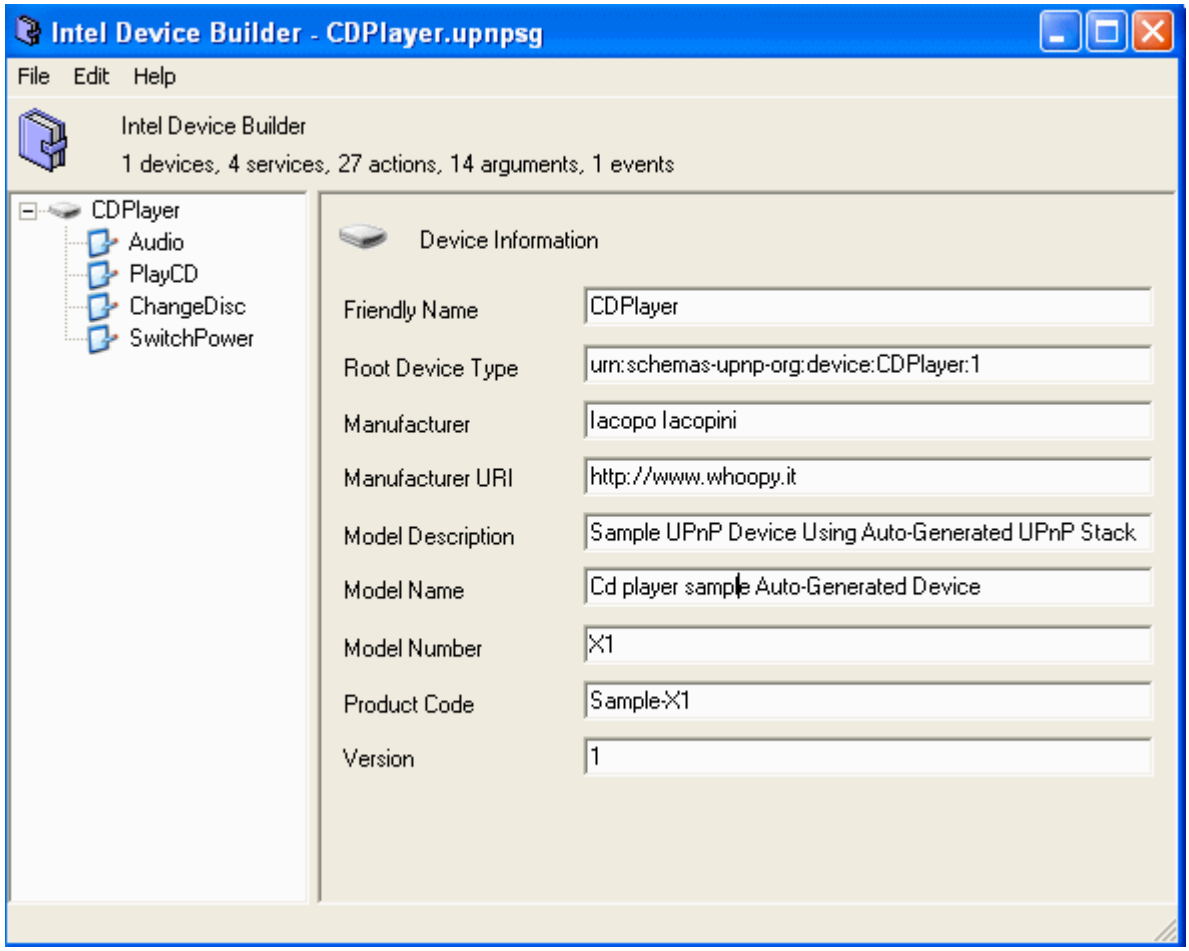


Figura 32: Schermata del tool Platform Builder dopo la definizione del dispositivo con tutti i suoi servizi

6.2.2.3. Descrizione XML del dispositivo

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType> urn:schemas-upnp-org:device:CDPlayer:1</deviceType>
    <friendlyName>CDPlayer</friendlyName>
    <manufacturer>Iacopo Iacopini</manufacturer>
    <manufacturerURL>http://www.whoopy.it</manufacturerURL>
    <modelDescription>Sample UPnP Device Using Auto-Generated UPnP Stack</modelDescription>
    <modelName>Cd player sample Auto-Generated Device</modelName>
    <modelNumber>X1</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:SwitchPower</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:ChangeDisc:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:ChangeDisc</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:PlayCD:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:PlayCD</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:Audio:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:Audio</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
```

```
      <controlURL>URL for control</controlURL>
      <eventSubURL>URL for eventing</eventSubURL>
     </service>
    </serviceList>
   <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

6.2.2.4. Funzionamento teorico

Per permettere all'utente la gestione delle variabili di stato del dispositivo, un control point dovrà invocare delle azioni appropriate che agiscono sulle variabili interessate.

```
Invoke SetTarghet with TRUE on SwitchPower service

Invoke AddDisc on ChangeDisc service
```

Figura 33: Codice di esempio per accendere il dispositivo e per permettergli d iaccettare un nuovo disco.

```
Check value of evented DoorIsOpen variable

Invoke CloseDoor

Invoke Play on PlayCD service
```

Figura 34: Codice di esempio per l'esecuzione automatica di un CD dopo il suo inserimento all'interno dell'alloggiamento.

```
Invoke SetVolume with new value on Audio
```

Figura 35: Codice di esempio per il settaggio del volume da parte di un Control Point.

6.3. Codice sorgente generato

Una volta terminata la fase di definizione del dispositivo mediante Intel Device Builder è possibile esportare in modo automatico il codice sorgente da completare cliccando su "Export Device Stack...". Vengono così generati in tutto 11 files di cui uno, chiamato *UPnPSample.vcproj,* è il file di progetto per il Microsoft Visual Studio .NET V7. Un secondo file molto importante è il *Main.c* all'interno del quale sono definite tutte le funzioni che agiscono sulle variabili di stato. Ogni funzione prevede uno statement di codice del tipo : *"/* TODO: Place Action Code Here... */"*. Al posto di queste righe il programmatore può scrivere il codice per personalizzare l'esecuzione dell'azione.

```
/* UPnP Device Main Module */

#ifndef MICROSTACK_NO_STDAFX
#include "stdafx.h"
#endif
#include <stdio.h>
#include <malloc.h>
#include <memory.h>
#include "ILibParsers.h"
```

```
#include "UPnPMicroStack.h"

void *UPnPmicroStackChain;
void *UPnPmicroStack;

DWORD UPnPMonitorSocketReserved;
WSAOVERLAPPED UPnPMonitorSocketStateObject;
SOCKET UPnPMonitorSocket;

void UPnPAudio_SetVolume(void* upnptoken,unsigned char NewVolume)
{
printf("Invoke: UPnPAudio_SetVolume(%u);\r\n",NewVolume);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_SetVolume(upnptoken);
}

void UPnPAudio_AddVolume(void* upnptoken,unsigned char NewVolume)
{
printf("Invoke: UPnPAudio_AddVolume(%u);\r\n",NewVolume);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_AddVolume(upnptoken,250);
}

void UPnPAudio_SetBalance(void* upnptoken,unsigned char NewBalance)
{
printf("Invoke: UPnPAudio_SetBalance(%u);\r\n",NewBalance);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_SetBalance(upnptoken);
}

void UPnPAudio_SetTreble(void* upnptoken,unsigned char NewTreble)
{
printf("Invoke: UPnPAudio_SetTreble(%u);\r\n",NewTreble);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_SetTreble(upnptoken);
}

void UPnPAudio_SetBass(void* upnptoken,unsigned char NewBass)
{
printf("Invoke: UPnPAudio_SetBass(%u);\r\n",NewBass);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_SetBass(upnptoken);
}

void UPnPAudio_SetFade(void* upnptoken,unsigned char NewFade)
{
printf("Invoke: UPnPAudio_SetFade(%u);\r\n",NewFade);
```

```
/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_Audio_SetFade(upnptoken);
}

void UPnPPlayCD_GetPlayProgram(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_GetPlayProgram();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_GetPlayProgram(upnptoken);
}

void UPnPPlayCD_GetDiscInfo(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_GetDiscInfo();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_GetDiscInfo(upnptoken,"Sample String");
}

void UPnPPlayCD_SetPlayProgram(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_SetPlayProgram();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_SetPlayProgram(upnptoken);
}

void UPnPPlayCD_GetTrackInfo(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_GetTrackInfo();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_GetTrackInfo(upnptoken,25000,250);
}

void UPnPPlayCD_Play(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_Play();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_Play(upnptoken);
}

void UPnPPlayCD_Stop(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_Stop();\r\n");

/* TODO: Place Action Code Here... */
```

```
/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_Stop(upnptoken);
}

void UPnPPlayCD_GetPlayMode(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_GetPlayMode();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_GetPlayMode(upnptoken);
}

void UPnPPlayCD_Pause(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_Pause();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_Pause(upnptoken);
}

void UPnPPlayCD_NextTrack(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_NextTrack();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_NextTrack(upnptoken);
}

void UPnPPlayCD_SelectTrack(void* upnptoken,unsigned char Number)
{
printf("Invoke: UPnPPlayCD_SelectTrack(%u);\r\n",Number);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_SelectTrack(upnptoken);
}

void UPnPPlayCD_PrevTrack(void* upnptoken)
{
printf("Invoke: UPnPPlayCD_PrevTrack();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_PlayCD_PrevTrack(upnptoken);
}

void UPnPChangeDisc_CloseDoor(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_CloseDoor();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
```

```
UPnPResponse_ChangeDisc_CloseDoor(upnptoken);
}

void UPnPChangeDisc_HasTrayDisc(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_HasTrayDisc();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_HasTrayDisc(upnptoken,1);
}

void UPnPChangeDisc_AddDisc(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_AddDisc();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_AddDisc(upnptoken);
}

void UPnPChangeDisc_NextDisc(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_NextDisc();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_NextDisc(upnptoken);
}

void UPnPChangeDisc_ToggleDoor(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_ToggleDoor();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_ToggleDoor(upnptoken);
}

void UPnPChangeDisc_PrevDisc(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_PrevDisc();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_PrevDisc(upnptoken);
}

void UPnPChangeDisc_IsDoorOpen(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_IsDoorOpen();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_IsDoorOpen(upnptoken,1);
}
```

```
void UPnPChangeDisc_RandomDisc(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_RandomDisc();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_RandomDisc(upnptoken);
}

void UPnPChangeDisc_OpenDoor(void* upnptoken)
{
printf("Invoke: UPnPChangeDisc_OpenDoor();\r\n");

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_ChangeDisc_OpenDoor(upnptoken);
}

void UPnPSwitchPower_SetTarghet(void* upnptoken,int State)
{
printf("Invoke: UPnPSwitchPower_SetTarghet(%d);\r\n",State);

/* TODO: Place Action Code Here... */

/* UPnPResponse_Error(upnptoken,404,"Method Not Implemented"); */
UPnPResponse_SwitchPower_SetTarghet(upnptoken);
}

void UPnPPresentationRequest(void* upnptoken, struct packetheader *packet)
{
printf("UPnP Presentation Request: %s %s\r\n", packet->Directive,packet->DirectiveObj);

/* TODO: Add Web Response Code Here... */
printf("HOST: %x\r\n",UPnPGetLocalInterfaceToHost(upnptoken));

UPnPPresentationResponse(upnptoken, "HTTP/1.0 200 OK\r\n\r\n" , 19 , 1);
}

DWORD WINAPI Run(LPVOID args)
{
getchar();
ILibStopChain(UPnPmicroStackChain);
return 0;
}

void CALLBACK UPnPIPAddressMonitor(
IN DWORD dwError,
IN DWORD cbTransferred,
IN LPWSAOVERLAPPED lpOverlapped,
IN DWORD dwFlags
)
{
UPnPIPAddressListChanged(UPnPmicroStack);
orSocket,SIO_ADDRESS_LIST_CHANGE,NULL,0,NULL,0,&UPnPMonitorSocketReserved,&UPnPMonitorSocketStateObject,&UP
}
int _tmain(int argc, _TCHAR* argv[])
{
UPnPmicroStackChain = ILibCreateChain();
```

```
/* TODO: Each device must have a unique device identifier (UDN) */
microStack = UPnPCreateMicroStack(UPnPmicroStackChain,"CDPlayer","3fb920a0-97e4-4f2b-8032-f08794a02806","0000001",18

UPnPFP_PresentationPage=&UPnPPresentationRequest;
UPnPFP_ChangeDisc_CloseDoor=&UPnPChangeDisc_CloseDoor;
UPnPFP_ChangeDisc_HasTrayDisc=&UPnPChangeDisc_HasTrayDisc;
UPnPFP_ChangeDisc_AddDisc=&UPnPChangeDisc_AddDisc;
UPnPFP_ChangeDisc_NextDisc=&UPnPChangeDisc_NextDisc;
UPnPFP_ChangeDisc_ToggleDoor=&UPnPChangeDisc_ToggleDoor;
UPnPFP_ChangeDisc_PrevDisc=&UPnPChangeDisc_PrevDisc;
UPnPFP_ChangeDisc_IsDoorOpen=&UPnPChangeDisc_IsDoorOpen;
UPnPFP_ChangeDisc_RandomDisc=&UPnPChangeDisc_RandomDisc;
UPnPFP_ChangeDisc_OpenDoor=&UPnPChangeDisc_OpenDoor;
UPnPFP_PlayCD_GetPlayProgram=&UPnPPlayCD_GetPlayProgram;
UPnPFP_PlayCD_GetDiscInfo=&UPnPPlayCD_GetDiscInfo;
UPnPFP_PlayCD_SetPlayProgram=&UPnPPlayCD_SetPlayProgram;
UPnPFP_PlayCD_GetTrackInfo=&UPnPPlayCD_GetTrackInfo;
UPnPFP_PlayCD_Play=&UPnPPlayCD_Play;
UPnPFP_PlayCD_Stop=&UPnPPlayCD_Stop;
UPnPFP_PlayCD_GetPlayMode=&UPnPPlayCD_GetPlayMode;
UPnPFP_PlayCD_Pause=&UPnPPlayCD_Pause;
UPnPFP_PlayCD_NextTrack=&UPnPPlayCD_NextTrack;
UPnPFP_PlayCD_SelectTrack=&UPnPPlayCD_SelectTrack;
UPnPFP_PlayCD_PrevTrack=&UPnPPlayCD_PrevTrack;
UPnPFP_SwitchPower_SetTarghet=&UPnPSwitchPower_SetTarghet;
UPnPFP_Audio_SetVolume=&UPnPAudio_SetVolume;
UPnPFP_Audio_AddVolume=&UPnPAudio_AddVolume;
UPnPFP_Audio_SetBalance=&UPnPAudio_SetBalance;
UPnPFP_Audio_SetTreble=&UPnPAudio_SetTreble;
UPnPFP_Audio_SetBass=&UPnPAudio_SetBass;
UPnPFP_Audio_SetFade=&UPnPAudio_SetFade;


/* All evented state variables MUST be initialized before UPnPStart is called. */
UPnPSetState_Audio_Volume(UPnPmicroStack,250);

printf("Intel MicroStack 1.0 - CDPlayer\r\n\r\n");
CreateThread(NULL,0,&Run,NULL,0,NULL);

UPnPMonitorSocket = socket(AF_INET,SOCK_DGRAM,0);
orSocket,SIO_ADDRESS_LIST_CHANGE,NULL,0,NULL,0,&UPnPMonitorSocketReserved,&UPnPMonitorSocketStateObject,&UP

ILibStartChain(UPnPmicroStackChain);

closesocket(UPnPMonitorSocket);
return 0;
}
```

Figura 36: Listato del file Main.h esportato dal Device Builder.

Del codice generato al programmatore è rimandato il compito di personalizzare sia l'esecuzione delle funzioni sia la gestione degli errori.

BIBLIOGRAFIA

Articoli e Relazioni

- "Open Services Gateway Initiative, specification overview", The OSGi (2000)
- "Home networking: il punto di vista di un operatore di telecomunicazioni", Paolo Pastorino, Giancarlo Lasagna (CSELT)
- "Home networking middleware", Amin Dhir (XILINX)
- "Univeral Plug And Play Device Architecture", Microsoft Corp.
- "Evolution of the Residential-Gateway Concept and Standards", Hongjun Li (Parks Associated)
- "Low-cost multi-service home gateway creates new business opportunities. White Paper"
- "Overview of residential gateways", Hongjun Li (Parks Associated)
- "The residential gateways: a market overview", Hongjun Li & Brian Canny (Parks Associated)
- "Scenari sul futuro: Domotica", (Telecom Italia Labs)
- "La domotica, Internet e gli hackers", Franco Guadagni (Telecom Italia Labs)
- "Cos'è una Home & Office Netwrk?", (Telecom Italia Labs)
- "How multiple services work on an OSGi service platform", John R. Barr (president of OSGi Alliance)
- "OSGi – Open Services Gateway Initiative", Timo Honkanen
- "Design Challenges for Home Gateway Devices", Satish Gupta (WIPRO)
- "Residential Gateways", Amin Dihr (XILINX)
- "Domotica: tecnologie e applicazioni per la casa", Anna Rossi (PCMagazine)
- "Intel Authoring Tools for UPnP Technologies", Research & development at Intel
- "Intel Development Tools for Implementing UPnP Devices", Research & development at Intel
- "Universal Plug and Play in Windows XP" – Microsoft TechNet