|  |
| --- |
| **LCL Guida ufficiale BETA 3.0.0 rev 12-01-2010** |
| Light Command Line beta 3.0.0 rev 9-1-2010 |
|  |
| Guida per l'utilizzo di LCL, software per registrare o modificare i file creati da LightWall BETA 17.3.3 e 17.3.4 Risolto bug sulla gestione del percorso !!! |
|  |
| **Ks89** |
| **12/01/2010** |
|  |

# attenzione versione 3.0.0 perfetta *SOLO* con 17.3.4

# NON COMPATIBILE CON 17.3.5, per questa serve x forza 3.0.1 che però anch'essa non è compatibile al 100%. infatti può solo ripristinare i file e modificare tutti i campi eccetto "codice13" Inoltre le registrazioni manuali ed automatiche non funzionano + perchè ora c'è di mezzo un controllo su codice13 NECESSARIO FARE AL PIù PRESTO 3.1.0 1. Descrizione struttura, comando e valore

## 1.1 Struttura

## 1.2 Comando

## 1.3 Valore

## 2. Tutte le strutture e i comandi possibili 2.1 Struttura "write"

## 2.2 Struttura "lic"

## 2.3 Struttura "writepc"

## 2.4 Struttura "pathw"

## 2.5 Struttura "license"

## 2.6 Struttura "save"

## 2.7 Struttura "restore"

## 2.8 Struttura "view"

## 2.9 Struttura "viewf"

## 2.A Struttura "generate"

## 2.B Struttura "simulate"

## 2.C Comando speciale: "matrice"

# 3. Eseguire una registrazione manuale parziale o completa diversa da simulate

# 1 Descrizione struttura, comando e valore

## 1.1 Struttura

Struttura è il campo dalla quale si diramano i comandi. Ognuna di esse possiede solo una serie limitata di comandi possibili.

## 1.2 Comando

Comando è il sottolivello di struttura. Ne esistono una serie limitata con particolari funzionalità ed applicazioni, indicate successivamente

## 1.3 Valore

Ultimo campo. In base alla struttura e al valore diventa il valore stesso nel file. Cioè il valore inserito viene copiato all'interno del campo della "struct del file" per poi essere salvato con l'apposita struttura "save".

# 2 Tutte le strutture e i comandi possibili 2.1 Struttura "write"

**write** : struttura associata al file ks.dat  
**struttura.comando. valore**

write.codice ( 11 numeri al massimo 0->10)

write.controllo (inserire codice, controllo viene generato in automatico)

write.reg (jhfiusq (registrato) o jhfiusw(non registrato))

write.inutile (5 numeri : 12344 (file non corrotto))

write.inutile2 (inserire 'y' o 'n'. Se y=-546,451405 cioè file NON corrotto)

write.pcname (al massimo di 20 caratteri)

write.space (inutile, massimo 2 caratteri)

write.path (al massimo 100 caratteri )

write.controllo\_registrazione (se 22: non registrato, se 32: registrato)

write.controlloic (30 numeri al massimo 0->29)

write.mixer (sempre e solo 1485170210, il prog separa i numeri in 1 4 8 5 1 7 0 2 10)  
write.hash (qualunque valore. md5 e sha-1 vengono generati in automatico e non in base al valore inserito)

2.2 Struttura "lic"

**lic** : struttura associata al file lic.dat  
**struttura.comando. valore**

lic.codice\_dat (11 numeri al massimo 0->10. DEVE ESSERE UGUALE A write.codice)

lic.controllo\_dat (inserire codice\_dat. DEVE ESSERE UGUALE A write.codice)

lic.reg (1234667 (registrato) o 1234567(non registrato))

lic.inutile (5 numeri : 12344 (file non corrotto))

lic.inutile2 (inserire 'y' o 'n'. Se y=-546,451405 cioè file NON corrotto)

lic.pcname (al massimo di 20 caratteri)

lic.controllo\_registrazione (se 22: non registrato, se 32: registrato)

lic.controlloic (30 numeri al massimo 0->29. Generato a caso in LightWall)

lic.codice\_a (7 numeri al massimo 0->9. Nessuna condizione\*)

lic.controllo\_a (inserire codice\_a. Nessuna condizione)  
lic.tipolic (se inserisci 'p' è PROF, se metto il numero 1->3 sono gli anni)  
lic.ora\_r (inserire ora registrazione "gg/mm/aaaa oo:mm:ss")  
lic.ora\_s (inserire ora scadenza "gg/mm/aaaa oo:mm:ss")  
lic.version (inserire stringa con questa forma x.x.x)  
lic.hash (qualunque valore. md5 e sha-1 vengono generati in automatico e non in base al valore inserito)

*\*Il programma genera in automatico un codice valido cercando di tenere il più possibile i valori inseriti dall'utente. Spesso però li deve sostituire per far si che possa funzionare.*  
2.3 Struttura "writepc"  
**writepc** : struttura associata al file rek.dll  
**struttura.comando. valore**writepc.pcname (al massimo di 20 caratteri)  
writepc.controlloic (7 numeri al massimo 0->9)  
writepc.hash (qualunque valore. md5 e sha-1 vengono generati in automatico e non in base al valore inserito)  
  
2.4 Struttura "pathw"  
**pathw** : struttura associata al file irr.dat  
**struttura.comando. valore**pathw.path (al massimo 100 caratteri )  
pathw.controlloic (7 numeri al massimo 0->9)  
pathw.cont\_licenza (valore da 0->3. Indica il numero di tentativi fatti senza successo per attivare la licenza. Questo dato viene incrementato fino a 3 dopo 3 errori alla domanda della licenza. A 3 LightWall richiede lo sblocco licenza)

pathw.hash (qualunque valore. md5 e sha-1 vengono generati in automatico e non in base al valore inserito)  
2.5 Struttura "license"

**license** : struttura associata al file license.lic   
**struttura.comando. valore**

license.version (inserire stringa con questa forma x.x.x)

license.codice (11 numeri al massimo 0->10. Deve essere uguale a write.codice)

license.controllo (inserire codice, controllo viene generato in automatico\*)  
license.nome (Inserire nome a cui è associata la licenza [non ammessi spazi])

license.cogn (Inserire cognome a cui è associata la licenza [non ammessi spazi]))

license.soc (Inserire società a cui è associata la licenza [non ammessi spazi]))

license.mail (Inserire e-mail a cui è associata la licenza [non ammessi spazi]\*\*)

license.reg (jhfiusq (registrato) o jhfiusw(non registrato)\*\*\*)

license.inutile (5 numeri : 12344 (file non corrotto))

license.inutile2 (inserire 'y' o 'n'. Se y=-546,451405 cioè file NON corrotto)

license.num\_domanda (numero da 1 a 6)

license.domanda (risposta alla domanda [50 caratteri al massimo. Spazi ammessi])

license.tipolic (se inserisci 'p' è PROF, se metto il numero 1->3 sono gli anni)  
license.ora\_r (inserire ora registrazione "gg/mm/aaaa oo:mm:ss")  
license.ora\_s (inserire ora scadenza "gg/mm/aaaa oo:mm:ss")  
license.hash (qualunque valore. md5 e sha-1 vengono generati in automatico e non in base al valore inserito)

*\** *Deve essere uguale a write.controllo  
\*\*Deve essere valida. Composta così: x@a.x  
\*\*\*Il valore deve essere uguale a quello di write.reg*

2.6 Struttura "save"

**save** : struttura che permette di salvare i file  
**struttura.comando. valore**  
save.ks (Qualunque. Il comando esegue il salvataggio dei dati precaricati)

save.rek (Qualunque. Il comando esegue il salvataggio dei dati precaricati)

save.irr (Qualunque. Il comando esegue il salvataggio dei dati precaricati)  
save.lic (Qualunque. Il comando esegue il salvataggio dei dati precaricati)  
save.license (Qualunque. Il comando esegue il salvataggio dei dati precaricati)  
save.all (Qualunque. Il comando esegue il salvataggio su tutti i file)

2.7 Struttura "restore"

**restore** : struttura che ripristina ai valori di fabbrica i file (sostituisce Ripristino fabbrica beta)  
**struttura.comando. valore**  
restore.ks (Qualunque. Il comando esegue il ripristino fabbrica del file indicato)

restore.rek (Qualunque. Il comando esegue il ripristino fabbrica del file indicato)

restore.lic (Qualunque. Il comando esegue il ripristino fabbrica del file indicato)

restore.all (Qualunque. Il comando esegue il ripristino fabbrica su tutti i file)

2.8 Struttura "view"  
**view** : struttura per visualizzare le informazioni contenute nella memoria del programma  
**struttura.comando. valore**view.ks (Mostra i dati nella struttura del programma)

view.rek (Mostra i dati nella struttura del programma)

view.irr (Mostra i dati nella struttura del programma)  
view.lic (Mostra i dati nella struttura del programma)  
view.license (Mostra i dati nella struttura del programma)

2.9 Struttura "viewf"  
**viewf** : struttura per visualizzare le informazioni contenute nei file  
**struttura.comando. valore**viewf.ks (Mostra i dati nella struttura del file)

viewf.rek (Mostra i dati nella struttura del file)

viewf.irr (Mostra i dati nella struttura del file)  
viewf.lic (Mostra i dati nella struttura del file)  
viewf.license (Mostra i dati nella struttura del file)

viewf.all (Carica i dati da tutti i file a tutte le struttura del programma\*)

N.B.  
*\*all, riempie tutti campi delle struct del programma con i dati letti dei file, sovrascrivendo tutti i valori non salvati precedentemente. E' un comando irreversibile.*

2.A Struttura "generate"  
**generate** : struttura per generare il codice di risposta  
**struttura.comando. valore**generate."qualunque" (Qualunque. Avvia la generazione del codice di risposta\*)

N.B.  
*\*E' la stessa generazione di generazione risposta beta 5. Da ora in poi si troverà solo qui*

2.B Struttura "simulate"  
**simulate** : struttura per avviare il wizard di registrazione guidata  
**struttura.comando. valore**simulate.registrazione (inserire ora registrazione "gg/mm/aaaa oo:mm:ss"\*)

simulate.complete (inserire qualunque valore\*\*)

N.B.  
*\*Questo comando genera in automatico una registrazione valida del programma impostando come data di registrazione quella attuale, mentre la scadenza quella impostata. Dalla differenza delle date capisce di che versione si tratti.  
Questo comando è molto limitato. Permette solo di registrare lightwall dopo la prima esecuzione, cioè non è in grado di cambiare il codice utente ecc..., ma modifica solo i campi su cui avviene il controllo per verificare se registrato o no.****PER POTERLO UTILIZZARE SEGUIRE ATTENTAMENTE QUESTA PROCEDURA:  
1) Eseguire un "restore.all.1" se già registrato  
2) Aprire e chiudere LightWall in modo che generi i vari codici  
3) Eseguire simulate.registrazione.gg/mm/aaaa oo:mm:ss***

*\*\*Questo comando esegue la registrazione di LIGHTWALL dal nulla, cioè, basta che vi siano i file necessari e LCL, tramite wizard, permette di creare una registrazione valida personalizzata del programma.  
Notare che con questo metodo posso inserire qualunque codice utente o percorsi, nomepc ecc..  
Il programma si occupa di rendere il codice licenza valido e anche le cifre di controllo.  
Il codice utente risulterà sempre valido perchè non dovrà creare un codice di risposta (questo processo viene bypassato). Con questo metodo l'utente ha molta libertà sul codice utente e soprattutto evita un procedimento che con una registrazione forzata come questa sarebbe davvero inutile  
NB:* ***utilizzare il percorso o nomepc "auto" per far si che venga inserito dirattamente nella struttura il percorso attuale e il nome del computer in funzione****. Questo procedimento funziona grazie alla lettura del percorso e nomepc all'avvio del programma.*

2.C Comando speciale: "matrice"

**matrice:** è un particolare comando, che non ha ne struttura ne valore, per generare la matrice su tutti i file.

xxxx.matrice.xxxx -> qualunque struttura e qualunque valore non influenzano la matrice.  
Essa viene generata in automatico dal programma dopo aver eseguito il comando view.all.1.  
Infatti, è importante che tutti i campi necessari siano stati pre-caricati. Essendo moltissimi, conviene usare il view.all.1

# 3. Eseguire una registrazione manuale parziale o completa diversa da simulate

ATTENZIONE: questo sistema esegue la registrazione mantenendo nomepc, path e vari codici.  
E' possibile cambiare tutto. Basta fare attenzione a rispettare le istruzioni date sopra.

**QUESTA PROCEDURA E' LA STESSA CHE FA SIMULATE.REGISTRAZIONE MA QUI SI PUO' FARE MANUALMENTE**

1) Posizionare i file "LCL beta 3.0.0.exe" e "libeay32.dll" nella cartella di installazione di LightWall  
2) Eseguire "LightWall.exe". Chiuderlo subito dopo.  
3) A questo punto si saranno generati i file aggiuntivi necessari.  
4) Eseguire LCL beta 3.0.0 e digitare i seguenti comandi in ordine

LISTA COMANDI:

>> viewf.all.1 //carica tutti i dati dai file al programma

>> viewf.ks.1 //mostra il contenuto di ks.dat

>> write.reg.jhfiusq //imposta reg=jhfiusq, cioè REGISTRATO

>> write.controllo\_registrazione.32 //imposta 32, cioè REGISTRATO

>> viewf.lic.1 //mostra il contenuto di lic.dat

>> lic.reg.1234667 //imposta 12345667, cioè REGISTRATO

>> lic.controllo\_registrazione.32 //imposta 32, cioè REGISTRATO

>> viewf.rek.1 //mostra il contenuto di rek.dll

>> save.ks.1 //salva ks.dat

>> save.rek.1 //salva rek.dll

>> save.lic.1 //salva lic.dat

>> a\_caso.matrice.1 //genera la matrice per tutti i file. vedi 2.6

>> save.ks.1 //salva ks.dat

>> save.rek.1 //salva rek.dll

>> save.lic.1 //salva lic.dat

>> exit //esce dal programma ('exit' o 'quit' causano l'uscita)  
  
**Per la prima volta, in beta 3.0.0 si può creare una registrazione completa sia tramite procedura guidata simulate.complete sia manualmente digitando i vari dati nei campi e salvando i file. Il procedimento è simile a quello sopra, ma nel caso di registrazione completa è necessario compilare accuratamente anche i campi dei codici, nomepc, percorso ecc... Essendo abbastanza lungo, simulate.complete esegue il procedimento in modo molto semplice e guidato.**