Novità di IBM Communications Server per Linux V6.2.1

Questo documento illustra le modifiche al rilascio di manutenzione per la versione 6.2.1.0 di Communications Server per Linux (Numero di prodotto 5724-i33, CS Linux) e di Communications Server per Linux in zSeries (Numero di prodotto 5724-i34, CS Linux in zSeries). Le descrizioni si applicano a entrambi i prodotti Communications Server per Linux e Communications Server per Linux in zSeries.

Le modifiche fornite in questo rilascio di manutenzione per Communications Server per Linux V6.2.1.0 comprendono:

1. Supporto kernel 2.6 - Supporto sistema operativo Linux aggiornato
2. Linux in Power - Supporto e impacchettamento piattaforma aggiuntivo
3. AIX Remote API Client - Supporto e impacchettamento aggiuntivo
4. Supporto LU primaria - Interfaccia di programmazione LUA per il supporto dei flussi LU 0 primari
5. Aggiornamenti a Remote API Client - Aggiornamenti forniti per i client
6. Aggiornamenti al manuale LUA Programmer's Guide - Interfaccia LU primaria aggiuntiva
7. Aggiornamenti al manuale Administration Guide - Comandi aggiuntivi per la definizione e l'interrogazione dei parametri del timer HPR
8. Aggiornamenti al manuale NOF Programmer's Guide - Istruzioni aggiuntive per la definizione e l'interrogazione dei parametri del timer HPR

Questa nuova funzione viene illustrata di seguito con ulteriori dettagli.

Supporto kernel 1.2.6

CS Linux e CS Linux in zSeries supportano adesso le installazioni server nelle distribuzioni kernel Linux 2.6 da Red Hat e SuSE. Le distribuzioni supportate per le installazioni server sono RHEL 4 e SLES 9-SP1. In fase di installazione del prodotto CS Linux nel sistema Intel, il kernel deve essere una distribuzione Linux a 32 bit. In fase di installazione di CS Linux in un sistema Linux in Power (OpenPower o Power5), il kernel deve essere una distribuzione Linux a 64 bit. Il prodotto CS Linux in zSeries può essere installato in un kernel Linux a 64 bit. Tuttavia, nel futuro, le distribuzioni Linux per la piattaforma zSeries saranno solo per il kernel a 64 bit. Le applicazioni possono utilizzare librerie a 32 bit o a 64 bit.

Per Remote API Client, che dispone del supporto della distribuzione kernel Linux 2.6, le applicazioni SNA client possono richiamare solo librerie a 32 bit nelle piattaforme Intel. Pertanto, se l'applicazione client è in esecuzione in una distribuzione Linux x86_64, possono essere utilizzate solo le librerie a 32 bit. Per le piattaforme Linux in zSeries e Linux in Power, le applicazioni SNA client possono utilizzare sia librerie a 32 sia a 64 bit.

Nella tabella seguente sono riportate le distribuzioni kernel Linux supportate nelle varie piattaforme:

Tabella 1. Distribuzioni kernel Linux supportate

Piattaforma	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
Intel						
Client (i686 - 32 bit)	x	x	x	x	x	
Client (x86_64), 32-bit API		x	x	x	x	
Server (i686 - 32 bit)	x	x	x	x	x	

Tabella 1. Distribuzioni kernel Linux supportate (Continua)

Piattaforma	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
OpenPower o Power 5 (kernel 64 bit, ppc64)						
Client				x	x	
Server				x	x	
Linux in zSeries (s390, 31-bit)						
Client		x	x	x	x	
Server		x	x	x*	x*	
Linux in zSeries (s390x, zSeries, 64 bit)						
Client		x	x	x	x	
Server		x	x	x	x	
AIX						
Client - 32 bit						x
Client - 64 bit						x

* indica che è possibile che il supporto non venga più fornito nei rilasci successivi di Linux.

2. Linux in Power

Il prodotto CS Linux (5724-i33) dispone adesso del supporto per più piattaforme che include il server CS Linux per le piattaforme Linux in Power, che sono dei server OpenPower e Power5 pSeries. In fase di installazione del server, il supporto per tali piattaforme richiede RHEL 4 o SLES 9-SP1. Per installare un server CS Linux in Linux in Power, fare riferimento a README.ppc64.xx_yy (in cui xx_yy è l'impostazione internazionale del sistema).

CS Linux Remote API client per Linux in Power viene fornito con i prodotti CS Linux e CS Linux per zSeries. Il nuovo supporto client si trova nella directory /ibm-commserver-clients/linux-ppc64 nel disco di installazione. Remote API client viene eseguito in Linux nello "Spazio utente" e non è kernel dipendente. Per installare il client, fare riferimento a /ibm-commserver-clients/linux-ppc64/README.

3. AIX Remote API Client

Quando Communications Servers viene eseguito in un dominio di server che supportano i Remote API client, fornisce adesso il supporto per le applicazioni SNA in esecuzione nelle piattaforme AIX V5. Durante la scrittura di applicazione per la piattaforma AIX, l'utente deve utilizzare la stessa descrizione per la compilazione e il collegamento utilizzata in Communications Server per AIX (http://www.ibm.com/software/network/commserver/library/aix).

La connessione tra il client AIX e i server Linux avverrà mediante TCP/IP. Un'applicazione SNA in AIX che utilizza CPI-C, APPC, LUA e alcune interfacce NOF per Communications Server per AIX V5 e successive, può utilizzare questo AIX Remote API client. Le precedenti interfacce CS/AIX V4.2 non sono supportate da Remote API Client.

4. Supporto LU principale

Le applicazioni LUA si connettono di solito ai mainframe host come LU secondarie. Pertanto la definizione per le sessioni sono controllate dall'applicazione host che invia il BIND per avviare una sessione. Communications Server può adesso agire come una LU principale per eseguire il downstream delle periferiche dipendenti SNA nelle interfacce LAN utilizzando l'interfaccia RUI_INIT_PRIMARY. Utilizzando tale interfaccia, un'applicazione può connettere sessioni LU dipendenti del downstream senza che sia necessario un mainframe host.

Per utilizzare le applicazioni LU principali, il nodo deve essere configurato con la LU di downstream (o un modello PU di downstream) che abbia un nome LU host uguale a #PRIRUI#. Ciò indicherà al server che le applicazioni che utilizzano RUI_INIT_PRIMARY controlleranno tali PU e le risorse LU ad esse assegnate. Le PU possono essere utilizzate solo nelle porte LAN. Fare riferimento agli aggiornamenti al manuale LUA Programming Guide seguenti, per le informazioni di interfaccia per la programmazione di applicazioni che utilizzano il supporto della LU primaria.

5. Aggiornamenti di Remote API Clients

5.1 Modifica al nome

Communications Server Remote API Client mostrerà adesso il nome ″IBM Remote API Client″ invece del nome precedente.

5.2 Installazione del client Linux e AIX

Se si sta installando il client o il server di Communications Server in un sistema che ha già una versione precedente del codice, è necessario disinstallare prima il server o il client. I file di configurazione non verranno eliminati o modificati dal processo di disinstallazione e di installazione.

5.3 Client Windows

Se si sta installando il client o il server di Communications Server in un sistema che ha già una versione precedente del codice, per mantenere la configurazione corrente è necessario eseguire i passi riportati di seguito:

- Arrestare il servizio client di Communications Server mediante il comando **net stop sxclient**. Questo comando deve essere immesso nel modo indicato (senza essere tradotto).
- Chiudere l'icona del monitor client di Windows (di solito si trova nella parte inferiore destra del desktop).
- Installare il prodotto senza rimuovere la versione precedente.

Se si rimuove il client installato precedente, le informazioni di configurazione vengono eliminate dal database del registro di Windows e devono essere immesse di nuovo dopo l'installazione dell'ultima versione.

IBM Remote API Client per Windows include adesso uno strumento di diagnostica per il servizio **snagetpd.exe**. Questo strumento crea un file compatto **snapd.exe**, che è auto-estraente. Il file di determinazione dei problemi contiene quanto segue:

- Contenuti del registro rilevanti
- Contenuti di directory o tutto quanto si trova nella directory e nelle sottodirectory di installazione
- Le informazioni della versione dei file di tutti i file binari installati
- L'ubicazione di tutti i file di traccia e di log
- L'emissione dai comandi 'ver', 'ipconfig /all', 'route print', 'netstat -an', 'netstat -a'

Lo strumento copia tutti i file di traccia e di log in snapd. Quando il servizio di Linux Communications Server specifica di fornire le informazioni di determinazione dei problemi, è necessario inviare un file snapd.exe da utilizzare come ausilio per l'assistenza dei problemi notificati.

Le fix APAR incluse per Windows Remote API Client sono:

- LI70604, LI70605 - L'installazione termina in modo anomalo in alcuni sistemi Windows XP in cui non è impostata una delle impostazioni internazionali predefinite.
- LI70677, LI70678 - Il thread principale dell'applicazione WinCPIC non può emettere più di un comando ALLOCATE.

6. Aggiornamenti al manuale LUA Programmer's Guide

Le informazioni seguenti sono complementari al manuale LUA Programmer's Guide per la descrizione dell'interfaccia di programma per RUI_INIT_PRIMARY. E' necessario fare riferimento alla sezione LUA Programmer's Guide per RUI_INIT per tutte le considerazioni o funzioni non descritte in modo esaustivo in questa sezione.

6.1 Progettazione e scrittura delle applicazioni LUA: Informazioni SNA per il principale RUI

Questa sezione contiene le considerazioni SNA per la scrittura di applicazioni principali RUI CS Linux per le comunicazioni con una LU di downstream.

Questa guida non spiega i concetti in modo particolareggiato. Se sono necessarie informazioni specifiche sui flussi dei messaggi SNA, fare riferimento alla documentazione per l'applicazione host per cui si sta progettando l'applicazione LUA di CS Linux.

6.2 Responsabilità dell'applicazione RUI principale

Un'applicazione RUI principale ha il controllo di entrambe le sessioni LU-SSCP e PLU-SLU a livello RU e può inviare e ricevere RU SNA in tali sessioni. La sessione PU-SSCP è interna di CS Linux e l'applicazione RUI principale non può accedervi.

Siccome un'applicazione RUI principale funziona a livello RU, ha un ampio margine di controllo sul flusso dei dati verso e dalla LU secondaria. Tuttavia, ha maggiori responsabilità rispetto ad un'applicazione LUA in quanto deve accertarsi che i messaggi SNA inviati siano validi e che i protocolli del livello RU (ad esempio il concatenamento e le parentesi) siano utilizzati correttamente. Nello specifico, si noti come CS Linux non verifica la validità delle RU inviate da un'applicazione RUI principale.

L'applicazione RUI principale si occupa di quanto segue:

- L'inizializzazione delle LU di downstream utilizzando RUI_INIT_PRIMARY e la relativa chiusura utilizzando RUI_TERM
- L'elaborazione dei messaggi NOTIFY dalla LU secondaria durante l'avvio e l'arresto delle applicazioni secondarie.
- L'elaborazione di INIT-SELF e TERM-SELF per attivare e disattivare la sessione PLU-SLU
- La creazione, invio, ricezione e analisi dei messaggi di flusso dei dati 3270 nelle RU dei dati.
- L'implementazione dei protocolli di livello RU (controllo richiesta, parentesi, concatenamento e verso)
- Crittografia (se necessaria)
- Compressione (se necessaria)

6.3 Restrizioni

CS Linux non supporta le seguenti applicazioni RUI principali:

- PU di downstream mediante DLUR

- DDDLU (Dynamically Defined Dependent LU)
- STSN di invio (per reimpostare i numeri della sequenza, l'applicazione deve eseguire l'UNBIND e di nuovo il BIND della sessione.

6.4 Informazioni di configurazione: Configurazione di collegamento RUI_INIT_PRIMARY

Per un'applicazione RUI principale che comunica con una LU di downstream, l'unica configurazione necessaria è quella della LU di downstream (o di un modello PU di downstream) che deve avere un nome LU host uguale a #PRIRUI#.

6.5 RUI_VERBS: Descrizione del controllo istruzione RUI_INIT_PRIMARY

L'istruzione RUI_INIT_PRIMARY stabilisce la sessione SSCP-LU per un'applicazione principale SNA che comunica con una LU di downstream. Se l'applicazione RUI agisce come applicazione secondaria SNA e comunica con una LU host, deve utilizzare RUI_INIT invece di RUI_INIT_PRIMARY.

6.5.1 Parametri forniti

L'applicazione fornisce i seguenti parametri (consultare /opt/ibm/sna/include/lua_c.h):

lua_verb
LUA_VERB_RUI

lua_verb_length
La lunghezza in byte del record dell'istruzione LUA. Impostare su sizeof(LUA_COMMON).

lua_opcode
LUA_OPCODE_RUI_INIT_PRIMARY

lua_correlator
LUA_OPCODE_RUI_INIT_PRIMARY

lua_luname
Il nome in ASCII della LU per cui avviare la sessione. Deve corrispondere al nome della LU di downstream configurata per l'utilizzo con il gateway SNA oppure ad una LU creata implicitamente da un modello LU di downstream con un nome LU host uguale a #PRIRUI#.

Questo parametro deve avere una lunghezza di 8 byte, riempito a destra con spazi, 0x20, se il nome è inferiore a 8 caratteri.

lua_max_length
La lunghezza di un buffer fornito per ricevere una copia della RU ACTLU(+RSP) ricevuta dalla PU di downstream. Se l'applicazione non deve ricevere queste informazioni, può specificare un puntatore nullo nel parametro lua_data_ptr e, in tal caso, non deve fornire un buffer di dati.

lua_data_ptr
Un puntatore per il buffer fornito per ricevere una copia della RU ACTLU(+RSP) ricevuta dalla PU di downstream. Se l'applicazione non deve ricevere queste informazioni, può specificare un puntatore nullo e le informazioni non verranno restituite.

lua_post_handle
Un puntatore per la routine di callback. Se l'istruzione si completa in modo asincrono, la LUA richiamerà questa routine per indicare il completamento dell'istruzione. Per ulteriori informazioni, consultare Progettazione e scrittura delle applicazioni LUA.

lua_encr_decr_option

0 La crittografia a livello di sessione non viene utilizzata.

128 Vengono eseguite la codifica e la decodifica.

Nota: La codifica e la decodifica sono eseguite dal programma dell'applicazione.

Un qualsiasi altro valore restituirà il codice di ritorno LUA_ENCR_DECR_LOAD_ERROR.

6.5.2 Parametri restituiti

LUA restituisce sempre il seguente parametro:

lua_flag2.async
Questo segnalatore è impostato su 1 se l'istruzione si completa in modo asincrono oppure su 0 se si completa in modo sincrono. RUI_INIT_PRIMARY viene completata sempre in modo asincrono salvo se restituisce un errore, come ad esempio, LUA_PARAMETER_CHECK.

Gli altri parametri restituiti dipendono dal completamento con esito positivo dell'istruzione; consultare le sezioni seguenti.

Esecuzione riuscita

Se l'istruzione ha esito positivo, la LUA restituisce i seguenti parametri.

lua_prim_rc
LUA_OK

lua_sid
Un ID sessione per la nuova sessione. Può essere utilizzato da istruzioni successive per identificare la sessione.

lua_data_length
La lunghezza della RU ACTLU(+RSP) ricevuta dalla PU di downstream. La LUA ubica i dati nel buffer specificato da lua_data_ptr.

Esecuzione non riuscita

Se l'istruzione ha esito negativo, la LUA restituisce un codice di ritorno principale per indicare il tipo di errore e un codice di ritorno secondario per fornire dettagli specifici sul motivo dell'esecuzione non riuscita.

Fare riferimento alla sezione LUA Programmer's Guide che descrive l'esecuzione non riuscita per RUI_INIT per consultare l'elenco dei possibili codici di ritorno. Inoltre, possono essere restituite le seguenti indicazioni per errori specifici RUI_INIT_PRIMARY:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT_PRIMARY

Un'istruzione RUI_INIT_PRIMARY è stata correntemente elaborata per questa sessione.

6.5.3 Interazione con altre istruzioni

L'istruzione RUI_INIT_PRIMARY deve essere la prima istruzione LUA emessa per la sessione.

Fino a quando questa istruzione non viene completata con esito positivo, l'unica altra istruzione LUA che può essere emessa per la sessione è RUI_TERM, che termina un'istruzione RUI_INIT_PRIMARY in sospeso.

Tutte le altre istruzioni emesse in questa sessione devono identificare la sessione utilizzando uno dei seguenti parametri da questa istruzione:

- L'ID sessione, restituito all'applicazione nel parametro lua_sid
- Il nome LU, fornito dall'applicazione nel parametro lua_luname

6.5.4 Utilizzo e restrizioni

L'istruzione RUI_INIT_PRIMARY che si completa dopo una risposta positiva ACTLU viene ricevuta dalla LU di downstream. Se necessario, l'istruzione attende all'infinito. Nel caso in cui l'applicazione deve controllare il contenuto di questa risposta positiva ACTLU, deve fornire un buffer di dati in RUI_INIT_PRIMARY (utilizzando i parametri lua_max_length e lua_data_ptr) in cui CS Linux restituisce il contenuto del messaggio ricevuto.

Una volta che l'istruzione RUI_INIT_PRIMARY viene completata correttamente, la sessione utilizza la LU per cui è stata avviata. Nessuna altra sessione LUA (da questa o da altre applicazioni) può utilizzare la LU fino a quando non viene emessa l'istruzione RUI_TERM o fino a quando non viene ricevuto un codice di ritorno principale LUA_SESSION_FAILURE.

Se l'istruzione RUI_INIT_PRIMARY restituisce un codice di ritorno principale LUA_IN_PROGRESS, l'ID sessione viene restituito nel parametro lua_sid. Questo ID sessione è lo stesso di quello restituito quando l'istruzione viene completata correttamente e può essere utilizzata con l'istruzione RUI_TERM per terminare un'istruzione RUI_INIT_PRIMARY in esecuzione.

7. Aggiornamenti al manuale Administration Guide

Vi sono due nuovi comandi per lo strumento di gestione snaadmin. Questi comandi sono:

define_rtp_tuning
definisce nuovi timer per l'ottimizzazione della connettività della sessione HPR.

query_rtp_tuning
esegue la query dei timer di commutazione del percorso HPR

Per utilizzare questi comandi, emettere il comando di aiuto ″snaadmin -d -h define_rtp_tuning″ oppure ″snaadmin -d -h query_rtp_tuning″ per visualizzare la sintassi dei comandi. E' possibile fare riferimento anche agli aggiornamenti al manuale NOF Programmer's Guide per ulteriori specifiche informazioni.

8. Aggiornamenti al manuale NOF Programmer's Guide

8.1 DEFINE_RTP_TUNING

DEFINE_RTP_TUNING specifica i parametri da utilizzare durante l'impostazione delle connessioni RTP. Dopo aver emesso questa istruzione, i parametri specificati verranno utilizzati per tutte le successive connessioni RTP fino a quando vengono modificati emettendo una nuova istruzione DEFINE_RTP_TUNING.

8.1.1 Parametri forniti

L'applicazione fornisce i seguenti parametri (consultare la struttura define_rtp_tuning in /opt/ibm/sna/include/nof_c.h):

opcode
AP_DEFINE_RTP_TUNING

path_switch_attempts
Il numero di tentativi di commutazione percorso per l'impostazione di nuove connessioni RTP. Specificare un valore compreso tra 1 e 255. Se si specifica zero, CS Linux utilizza il valore predefinito 6.

short_req_retry_limit
Il numero di volte per cui viene inviata una richiesta di stato prima che CS Linux determina che una connessione RTP è stata terminata ed avvia l'elaborazione della commutazione percorso. Specificare un valore compreso tra 1 e 255. Se si specifica zero, CS Linux utilizza il valore predefinito 6.

path_switch_times
La durata in secondi per cui CS Linux tenta di commutare un percorso di una connessione RTP

interrotta. Questo parametro viene specificato come quattro limiti di tempo separati per ciascuna delle priorità di trasmissione valide nell'ordine: AP_LOW, AP_MEDIUM , AP_HIGH e AP_NETWORK. Ciascuna di queste voci deve essere compresa nell'intervallo 1-65535. Il valore specificato per ciascuna priorità di trasmissione non deve superare il valore di una qualsiasi priorità di trasmissione inferiore.

Se si specifica 0 (zero) per uno di questi valori, CS Linux utilizza il valore predefinito corrispondente, nel modo riportato di seguito:

- 480 secondi (8 minuti) per AP_LOW
- 240 secondi (4 minuti) per AP_MEDIUM
- 120 secondi (2 minuti) per AP_HIGH
- 60 secondi (1 minuto) per AP_NETWORK

8.1.2 Parametri restituiti

Esecuzione riuscita

Se l'istruzione ha esito positivo, CS Linux restituisce i seguenti parametri:

primary_rc
AP_OK

Esecuzione non riuscita

Se l'istruzione non viene eseguita per un errore di parametro, CS Linux restituisce i seguenti parametri:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
I valori possibili sono:

AP_INVALID_PATH_SWITCH_TIMES Il parametro path_switch_times non è valido; ad esempio, è stato specificato un valore per una priorità di trasmissione che supera il valore specificato per una priorità di trasmissione inferiore.

Codici di ritorno comuni elenca ulteriori codici di ritorno secondari associati a AP_PARAMETER_CHECK, che sono comuni a tutte le istruzioni NOF.

8.2 QUERY_RTP_TUNING

QUERY_RTP_TUNING restituisce le informazioni relative ai parametri che verranno utilizzati per le successive connessioni RTP. Queste informazioni erano state impostate precedentemente utilizzando DEFINE_RTP_TUNING.

8.2.1 Parametri forniti

L'applicazione fornisce i seguenti parametri (consultare la struttura query_rtp_tuning in /opt/ibm/sna/include/nof_c.h):

opcode
AP_QUERY_RTP_TUNING

8.2.2 Parametri restituiti

Esecuzione riuscita

Se l'istruzione ha esito positivo, CS Linux restituisce i seguenti parametri:

primary_rc
AP_OK

path_switch_attempts
Il numero di tentativi di commutazione percorso per l'impostazione di nuove connessioni RTP

short_req_retry_limit
Il numero di volte per cui viene inviata una richiesta di stato prima che CS Linux determina che una connessione RTP è stata terminata ed avvia l'elaborazione della commutazione percorso.

path_switch_times
La durata in secondi per cui CS Linux tenta di commutare un percorso di una connessione RTP interrotta. Questo parametro viene specificato come quattro limiti di tempo separati per ciascuna delle priorità di trasmissione valide nell'ordine: AP_LOW, AP_MEDIUM , AP_HIGH e AP_NETWORK.

Altre condizioni

Codici di ritorno comuni elenca ulteriori combinazioni di codici di ritorno principali e secondari che sono comuni a tutte le istruzioni NOF.