O Que Há de Novo no IBM Communications Server para Linux V6.2.1

Este documento descreve as alterações de manutenção do release para a versão 6.2.1.0 do Communications Server para Linux (No. de Produto 5724-i33, CS Linux) e o Communications Server para Linux no zSeries (No. de Produto 5724-i34, CS Linux no zSeries). As descrições se aplicam aos produtos Communications Server para Linux e Communications Server para Linux no zSeries.

As alterações fornecidas neste release de manutenção do Communications Server para Linux V6.2.1.0 incorporam:

1. Suporte ao Kernel 2.6 - Suporte atualizado para o sistema operacional Linux
2. Linux on Power - Suporte e pacote adicionais de plataforma
3. Remote API Client do AIX - Plataforma e pacote adicional
4. Suporte a LU Primária - Interface de programação de LUA para suportar fluxos de LU Primária 0
5. Atualizações do Remote API Client - Atualizações fornecidas para os clientes
6. Atualizações do Guia do Programador da LUA - Interface adicional de LU Primária
7. Atualizações do Guia de Administração - Comandos adicionais para definir e consultar parâmetros de cronômetro HPR
8. Atualizações do Guia do Programador NOF - Verbos adicionais para definir e consultar parâmetros de cronômetro HPR

Essa nova função é descrita mais detalhadamente a seguir.

1. Suporte ao Kernel 2.6

O CS Linux e o CS Linux on zSeries agora suportam instalações do servidor nas distribuições do kernel 2.6 do Linux do Red Hat e SuSE. As distribuições suportadas para as instalações de servidor são RHEL 4 e SLES 9-SP1. Ao instalar o produto CS Linux num sistema Intel, o kernel deve ser uma distribuição Linux de 32 bits. Ao instalar o CS Linux em um sistema Linux on Power (OpenPower ou Power5), o kernel deve ser uma distribuição Linux de 64 bits. O produto CS Linux on zSeries pode ser instalado em um kernel Linux de 31 ou 64 bits. Entretanto, no futuro, as distribuições Linux para a plataforma zSeries serão feitas somente para o kernel de 64 bits. Os aplicativos podem utilizar bibliotecas de 32 ou 64 bits.

Para o Remote API Client, que tinha suporte para distribuição do kernel 2.6 do Linux, os aplicativos SNA cliente podem chamar somente bibliotecas de 32 bits em plataformas Intel. Dessa forma, se o aplicativo cliente estiver sendo executado em uma distribuição Linux x86_64, somente as bibliotecas de 32 bits poderão ser utilizadas. Para plataformas Linux on zSeries e Linux on Power, os aplicativos SNA cliente podem utilizar bibliotecas de 32 ou 64 bits.

As distribuições do kernel Linux suportadas nas diversas plataformas são mapeadas na tabela a seguir:

Tabela 1. Distribuições Suportadas do kernel Linux

Plataforma	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
Intel						
Cliente (i686 - 32 bits)	x	x	x	x	x	
Cliente (x86_64), API de 32 bits		x	x	x	x	

Tabela 1. Distribuições Suportadas do kernel Linux (continuação)

Plataforma	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
Servidor (i686 - 32 bits)	x	x	x	x	x	
OpenPower ou Power 5 (kernel de 64 bits, ppc64)						
Cliente				x	x	
Servidor				x	x	
Linux on zSeries (s390, 31 bits)						
Cliente		x	x	x	x	
Servidor		x	x	x*	x*	
Linux on zSeries (s390x, zSeries, 64 bits)						
Cliente		x	x	x	x	
Servidor		x	x	x	x	
AIX						
Cliente - 32 bits						x
Cliente - 64 bits						x

* indica suporte que pode ser desativado em futuros releases do Linux

2. Linux on Power

O produto CS Linux (5724-i33) agora possui suporte multiplataforma que inclui o servidor CS Linux para plataformas Linux on Power, que são o servidores OpenPower e os servidores Power5 pSeries. Ao instalar o servidor, o suporte para essas plataformas requer RHEL 4 ou SLES 9-SP1. Para instalar um servidor CS Linux em um Linux on Power, consulte README.ppc64.xx_yy (onde xx_yy é o código do idioma do sistema).

O CS Linux Remote API Client para Linux on Power é enviado com os produtos CS Linux e CS Linux para zSeries. O novo suporte a clientes está localizado no diretório /ibm-commserver-clients/linux-ppc64 no disco de instalação. O Remote API Client é executado no ″Espaço do Usuário″ no Linux e não depende do kernel. Para instalar o cliente, consulte /ibm-commserver-clients/linux-ppc64/README.

3. Remote API Client do AIX

Os servidores de comunicação que estejam em execução em um domínio de servidores que suportem Remote API Clients agora podem fornecer suporte para aplicativos SNA em execução nas plataformas AIX V5. Ao gravar aplicativos para a plataforma AIX, você deve utilizar a mesma descrição para compilar e vincular localizada no Communications Server para AIX (http://www.ibm.com/software/network/commserver/library/aix).

A conexão entre o cliente AIX e os servidores Linux será feito por meio de TCP/IP. Qualquer aplicativo SNA no AIX que utilize CPI-C, APPC, LUA e algumas interfaces NOF para o Communications Server para AIX V5 e superiores, pode utilizar esse Remote API Client do AIX. As interfaces CS/AIX V4.2 mais antigas não são suportadas pelo Remote API Client.

4. Suporte a LU Primária

Os aplicativos LUA normalmente se conectam a mainframes host como LUs secundárias. Isso significa que as definições das sessões são controladas pelo aplicativo host que envia BIND para iniciar uma sessão. O servidor de comunicação agora tem a habilidade de agir como uma LU Primária para recebimento de dados de dispositivos dependentes de SNA por interfaces LAN utilizando a interface RUI_INIT_PRIMARY. Ao utilizar essa interface, um aplicativo pode conectar sessões LU dependentes de recebimento de dados sem a necessidade de um mainframe host.

Para utilizar aplicativos de LU Primária, o nó deve estar configurado com LU de recebimento de dados (ou um gabarito de PU de Recebimento de Dados) que tenha o nome de LU host #PRIRUI#. Isso indicará ao servidor que os aplicativos que utilizem RUI_INIT_PRIMARY controlarão essas PUs e os recursos da LU designados a eles. As PUs podem ser utilizadas somente em portas LAN. Consulte as atualizações do Guia de Programação LUA (a seguir) para obter informações da interface para programar aplicativos para utilizar o suporte de LU Primária.

5. Atualizações de Remote API Clients

5.1 Alteração de Nome

O Communications Server Remote API Client agora mostrará o nome ″IBM Remote API Client″ no lugar do nome anterior.

5.2 Instalação do Cliente Linux e AIX

Se estiver instalando o servidor ou o cliente do servidor de comunicação em um sistema que já tenha uma versão anterior do código, você deverá primeiramente desinstalar o servidor ou o cliente. Os arquivos de configuração não serão excluídos nem modificados pelo processo de desinstalação e instalação.

5.3 Clientes Windows

Se estiver instalando o cliente do servidor de comunicação em um sistema que já tenha uma versão anterior do código, você deverá tomar as seguintes etapas para manter a configuração atual.

- Pare o serviço do cliente do servidor de comunicação emitindo **net stop sxclient**. Esse comando deve ser digitado como visto (não traduzido).
- Feche o ícone o monitor do cliente Windows (normalmente localizado na parte inferior direita do desktop).
- Instale o produto sem remover a versão anterior.

Se você remover o cliente Windows anterior que está instalado, as informações de configuração serão excluídas do banco de dados do registro do Windows e as informações de configuração deverão ser digitadas novamente depois que a versão mais recente for instalada.

O IBM Remote API Client para Windows agora inclui uma ferramenta de diagnóstico para o serviço, **snagetpd.exe**. Essa ferramenta cria um arquivo compactado, **snapd.exe**, de extração automática. O arquivo de determinação de problemas contém:

- Conteúdo relevante do registro
- Conteúdo do diretório de tudo no e sob o diretório de instalação
- Informações da versão do arquivo de todos os binários instalados

- Locais de todos os arquivos de rastreio e log
- Saída dos comando 'ver', 'ipconfig /all', 'route print', 'netstat -an' e 'netstat -a'

A ferramenta copia todos os arquivos de rastreio e log para snapd. Quando for direcionado pelo serviço Communications Server Linux a fornecer informações de determinação de problemas, você deverá enviar um arquivo snapd.exe. Ele auxiliará no fornecimento de serviço para os problemas relatados.

As correções APAR incluídas para o Remote API Client do Windows são:
- LI70604, LI70605 - A instalação é interrompida em alguns sistemas Windows XP que não têm um dos códigos de idioma padrão definidos.
- LI70677, LI70678 - O encadeamento principal do aplicativo WinCPIC não pode emitir mais de um ALLOCATE.

---

6. Atualizações do Guia do Programador LUA

As informações a seguir são aditivos para o Guia do Programador LUA para descrever a interface do programa para RUI_INIT_PRIMARY. Consulte a seção Guia do Programador LUA para RUI_INIT para obter quaisquer considerações ou recursos que possam não estar totalmente descritos nesta seção.

6.1 Projetando e Gravando Aplicativos LUA: Informações de SNA para RUI Primary

Essa seção contém considerações de SNA para gravar aplicativos CS Linux RUI Primary para comunicações com uma LU de recebimento de dados.

Esse guia não tenta explicar os conceitos do SNA em detalhes. Se você precisar de informações específicas sobre fluxos de mensagens SNA, consulte a documentação do aplicativo host para o qual está projetando seu aplicativo CS Linux LUA.

6.2 Responsabilidades do Aplicativo Primary RUI

Um aplicativo Primary RUI tem controle das sessões LU-SSCP e PLU-SLU no nível da RU (Request/Response Unit) e pode enviar e receber RUs SNA nessas sessões. A sessão PU-SSCP é interna ao CS Linux e o aplicativo Primary RUI não pode acessá-la.

Como um aplicativo Primary RUI funciona no nível da RU, ele possui um alto grau de controle sobre o fluxo de dados emitido e recebido da LU secundária. No entanto, ele tem uma responsabilidade maior que um aplicativo LUA comum para assegurar que as mensagens SNA enviadas por ele sejam válidas e que os protocolos do nível da RU (por exemplo, suporte e encadeamento) sejam utilizados corretamente. Especificamente, note que o CS Linux não tenta verificar a validade de RUs enviadas por um aplicativo Primary RUI.

O aplicativo Primary RUI é responsável por:
- Inicializar as LUs de recebimento de dados utilizando RUI_INIT_PRIMARY e finalizá-las utilizando RUI_TERM
- Processar mensagens NOTIFY a partir da LU secundária como início e parada de aplicativos secundários
- Processar INIT-SELF e TERM-SELF para ativar e desativar a sessão PLU-SLU
- Construir, enviar, receber e analisar mensagens de fluxo de dados 3270 em RUs de dados
- Implementar protocolos do nível da RU (controle de pedidos, suporte, encadeamento e direção)
- Criptografia (se necessária)
- Compactação (se necessária)

6.3 Restrições

O CS Linux não suporta os seguintes aplicativos Primary RUI:

- PUs de recebimento de dados sobre DLUR
- DDDLU (Dynamically Defined Dependent LUs)
- Envio de STSN (para reconfigurar números de seqüência, o aplicativo deve executar UNBIND e BIND novamente para a sessão).

6.4 Informações de Configuração: Configuração do Link RUI_INIT_PRIMARY

Para um aplicativo Primary RUI que esteja se comunicando com uma LU de recebimento de dados, a única configuração necessária é a LU de recebimento de dados (ou um gabarito de PU de Recebimento de Dados) que tenha o nome de LU host #PRIRUI#.

6.5 RUI_VERBS: Descrição de Controle do Verbo RUI_INIT_PRIMARY

O verbo RUI_INIT_PRIMARY estabelece a sessão SSCP-LU para um aplicativo Primário SNA que esteja se comunicando com uma LU de recebimento de dados. Se o aplicativo RUI agir como um SNA secundário e comunicar-se com uma LU host, ele deverá utilizar RUI_INIT em vez de RUI_INIT_PRIMARY.

6.5.1 Parâmetros Fornecidos

O aplicativo fornece os seguintes parâmetros (Consulte /opt/ibm/sna/include/lua_c.h):

lua_verb
LUA_VERB_RUI

lua_verb_length
O comprimento em bytes do registro do verbo LUA. Defina isso para sizeof(LUA_COMMON).

lua_opcode
LUA_OPCODE_RUI_INIT_PRIMARY

lua_correlator
LUA_OPCODE_RUI_INIT_PRIMARY

lua_luname
O nome, em ASCII, da LU para a qual você deseja iniciar a sessão. Deve corresponder ao nome de uma LU de recebimento de dados configurada para utilização com o Gateway SNA ou uma LU criada implicitamente a partir de um gabarito de LU de recebimento de dados que tenha o nome de LU host #PRIRUI#.

Esse parâmetro deve ter oito bytes de comprimento; preencha à direita com espaços, 0x20, se o nome tiver menos de oito caracteres.

lua_max_length
O comprimento de um buffer fornecido para receber uma cópia da RU ACTLU(+RSP) recebida da PU de recebimento de dados. Se o aplicativo não precisar receber essas informações, poderá especificar um ponteiro nulo no parâmetro lua_data_ptr, caso no qual não precisa fornecer um buffer de dados.

lua_data_ptr
Um ponteiro para o buffer fornecido para receber uma cópia da RU ACTLU(+RSP) recebida da PU de recebimento de dados. Se o aplicativo não precisar receber essas informações, poderá especificar um ponteiro nulo e as informações não serão retornadas.

lua_post_handle
Um ponteiro para uma rotina de retorno de chamada. Se o verbo for concluído de forma assíncrona, LUA irá chamar essa rotina para indicar a conclusão do verbo. Para obter informações adicionais, consulte Projetando e Gravando Aplicativos LUA.

lua_encr_decr_option

0 A criptografia do nível da sessão não é utilizada.

128 A criptografia e decriptografia são desempenhadas.

Nota: A criptografia e a decriptografia são desempenhadas pelo programa aplicativo.

Qualquer outro valor resultará no código de retorno LUA_ENCR_DECR_LOAD_ERROR.

6.5.2 Parâmetros Retornados

LUA sempre retorna o seguinte parâmetro:

lua_flag2.async
Esse sinalizador é definido como 1 se o verbo for concluído de forma assíncrona, ou 0 se o verbo for concluído de forma síncrona. RUI_INIT_PRIMARY sempre será concluído de forma assíncrona, a menos que retorne um erro como LUA_PARAMETER_CHECK.

Outros parâmetros retornados dependem da conclusão com êxito do verbo; consulte as seções a seguir.

Execução Com Êxito

Se o verbo for executado com êxito, LUA retornará os seguintes parâmetros.

lua_prim_rc
LUA_OK

lua_sid
Um ID de sessão para a nova sessão. Ele pode ser utilizado por verbos subseqüentes para identificar essa sessão.

lua_data_length
O comprimento da RU ACTLU(+RSP) recebida da PU de recebimento de dados. LUA coloca os dados no buffer especificado por lua_data_ptr.

Execução Sem Êxito

Se o verbo não for concluído com êxito, LUA retornará um código de retorno primário para indicar o tipo de erro e um código de retorno secundário para fornecer detalhes específicos sobre o motivo da execução sem êxito.

Consulte a seção Guia do Programador LUA que descreve a Execução Sem Êxito para RUI_INIT para consultar a lista de códigos de retorno possíveis. Além desses, as seguintes indicações podem ser retornadas para erros específicos de RUI_INIT_PRIMARY:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT_PRIMARY

Um verbo RUI_INIT_PRIMARY está sendo processado atualmente para essa sessão.

6.5.3 Interação com Outros Verbos

O verbo RUI_INIT_PRIMARY deve ser o primeiro verbo LUA emitido para a sessão.

Até que esse verbo seja concluído com êxito, o único verbo LUA adicional que pode ser emitido para essa seção é RUI_TERM, que finalizará um RUI_INIT_PRIMARY pendente.

Todos os demais verbos emitidos nessa sessão devem identificar a sessão utilizando um dos seguintes parâmetros a partir desse verbo:

- O ID de sessão, retornado para o aplicativo no parâmetro lua_sid
- O nome da LU, fornecido pelo aplicativo no parâmetro lua_luname

6.5.4 Uso e Restrições

O verbo RUI_INIT_PRIMARY é concluído após uma resposta positiva da ACTLU ser recebida da LU de recebimento de dados. Se necessário, o verbo aguardará indefinidamente. Se o aplicativo precisar verificar o conteúdo dessa resposta positiva ACTLU, poderá fazê-lo fornecendo um buffer de dados em RUI_INIT_PRIMARY (utilizando os parâmetros lua_max_length e lua_data_ptr), onde o CS Linux retorna o conteúdo da mensagem recebida.

Depois que o verbo RUI_INIT_PRIMARY for concluído com êxito, essa sessão utilizará a LU para a qual a sessão foi iniciada. Nenhuma outra sessão LUA (desse ou de nenhum outro aplicativo) pode utilizar a LU até que o verbo RUI_TERM seja emitido ou até que um código de retorno primário LUA_SESSION_FAILURE seja recebido.

Se o verbo RUI_INIT_PRIMARY retornar com um código de retorno primário LUA_IN_PROGRESS, o ID de sessão será retornado no parâmetro lua_sid. Esse ID de sessão é o mesmo que aquele retornado quando o verbo é concluído com êxito e pode ser utilizado com o verbo RUI_TERM para finalizar um verbo RUI_INIT_PRIMARY notável.

7. Atualizações do Guia de Administração

Há dois novos comandos para a ferramenta de administração snaadmin. Esses comandos são:

define_rtp_tuning
definir novos cronômetros para ajustar a conectividade da sessão HPR.

query_rtp_tuning
consultar cronômetros de comutador de caminho HPR

Para utilizar esses comandos, emita o comando de ajuda ″snaadmin -d -h define_rtp_tuning″ ou ″snaadmin -d -h query_rtp_tuning″ para consultar a sintaxe dos comandos. É possível, também, consultar as atualizações do Guia do Programador NOF para obter informações mais específicas.

8. Atualizações do Guia do Programador NOF

8.1 DEFINE_RTP_TUNING

DEFINE_RTP_TUNING especifica parâmetros a serem utilizados ao configurar conexões RTP. Depois de emitir esse verbo, os parâmetros especificados serão utilizados para todas as futuras conexões RTP, até que sejam modificados pela emissão de um novo verbo DEFINE_RTP_TUNING.

8.1.1 Parâmetros Fornecidos

O aplicativo fornece os seguintes parâmetros (Consulte a estrutura de define_rtp_tuning em /opt/ibm/sna/include/nof_c.h):

opcode
AP_DEFINE_RTP_TUNING

path_switch_attempts
Número de tentativas de comutação de caminho a serem definidas em novas conexões RTP. Especifique um valor no intervalo 1-255. Se você especificar 0 (zero), o CS Linux utilizará o valor padrão 6.

short_req_retry_limit
Número de vezes que um Pedido de Status é enviado antes que o CS Linux determine que uma conexão RTP está desconectada e inicie o processamento da Comutação de Caminho. Especifique um valor no intervalo de 1 a 255. Se você especificar 0 (zero), o CS Linux utilizará o valor padrão 6.

path_switch_times
Quantidade de tempo em segundos para o qual o CS Linux tenta comutar o caminho de uma conexão RTP desconectada. Esse parâmetro é especificado como quatro limites de tempo separados para cada uma das prioridades de transmissão válidas na ordem: AP_LOW, AP_MEDIUM, AP_HIGH e AP_NETWORK. Cada um deles deve estar no intervalo de 1 a 65535. O valor especificado para cada prioridade de transmissão não deve exceder o valor de nenhuma prioridade de transmissão inferior.

Se você especificar 0 (zero) para algum desses valores, o CS Linux utilizará o valor padrão correspondente da seguinte forma:

- 480 segundos (8 minutos) para AP_LOW
- 240 segundos (4 minutos) para AP_MEDIUM
- 120 segundos (2 minutos) para AP_HIGH
- 60 segundos (1 minuto) para AP_NETWORK

8.1.2 Parâmetros Retornados

Execução Com Êxito

Se o verbo for executado com êxito, o CS Linux retornará os seguintes parâmetros:

primary_rc
AP_OK

Execução Sem Êxito

Se o verbo não for executado devido a um erro de parâmetro, o CS Linux retornará os seguintes parâmetros:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Os valores possíveis são:

AP_INVALID_PATH_SWITCH_TIMES O parâmetro path_switch_times não era válido; por exemplo, é possível ter especificado um valor para uma prioridade de transmissão que exceda o valor especificado para uma prioridade de transmissão mais baixa.

Os Códigos de Retorno Comuns listam códigos de retorno secundário adicionais associados a AP_PARAMETER_CHECK, que são comuns a todos os verbos NOF.

8.2 QUERY_RTP_TUNING

QUERY_RTP_TUNING retorna informações sobre os parâmetros que serão utilizados para futuras conexões RTP. Essas informações eram configuradas anteriormente utilizando DEFINE_RTP_TUNING.

8.2.1 Parâmetros Fornecidos

O aplicativo fornece os seguintes parâmetros (Consulte a estrutura de query_rtp_tuning em /opt/ibm/sna/include/nof_c.h):

opcode
AP_QUERY_RTP_TUNING

8.2.2 Parâmetros Retornados

Execução Com Êxito

Se o verbo for executado com êxito, o CS Linux retornará os seguintes parâmetros:

primary_rc
AP_OK

path_switch_attempts
Número de tentativas de comutação de caminho a serem definidas em novas conexões RTP

short_req_retry_limit
Número de vezes que um Pedido de Status é enviado antes que o CS Linux determine que uma conexão RTP está desconectada e inicie o processamento da Comutação de Caminho.

path_switch_times
Quantidade de tempo em segundos para o qual o CS Linux tenta comutar o caminho de uma conexão RTP desconectada. Esse parâmetro é especificado como quatro limites de tempo separados para cada uma das prioridades de transmissão válidas na ordem: AP_LOW, AP_MEDIUM, AP_HIGH e AP_NETWORK.

Outras Condições

Os Códigos de Retorno Comuns listam combinações adicionais de códigos de retorno primários e secundários que são comuns a todos os verbos NOF.