Neuerungen in IBM Communications Server für Linux Version 6.2.1

Dieses Dokument beschreibt das Wartungsrelease mit Änderungen für Version 6.2.1.0 von Communications Server für Linux (Produktnummer 5724-i33, CS Linux) und Communications Server für Linux auf zSeries (Produktnummer 5724-i34, CS Linux auf zSeries). Die Beschreibungen gelten sowohl für Communications Server für Linux als auch für Communications Server für Linux auf zSeries.

Änderungen in diesem Wartungsrelease für Communications Server für Linux Version 6.2.1.0:

1. Unterstützung für Kernel 2.6 - Aktualisierte Unterstützung für das Betriebssystem Linux
2. Linux auf POWER - Zusätzliche Plattformunterstützung und neuer Paketinhalt
3. Remote API Client für AIX - Zusätzliche Plattform und neuer Paketinhalt
4. Unterstützung für primäre LU - Programmierschnittstelle LUA zur Unterstützung des Datenflusses für primäre LU 0
5. Updates zu Remote API Clients - Verfügbare Updates für die Clients
6. Updates zum LUA Programmer's Guide - Zusätzliche Schnittstelle für primäre LU
7. Updates zum Administratorhandbuch - Zusätzliche Befehle zum Definieren und Abfragen von Parametern des HPR-Zeitgebers
8. Updates zum NOF Programmer's Guide - Zusätzliche Verben zum Definieren und Abfragen von Parametern des HPR-Zeitgebers

Diese neuen Funktionen sind im Folgenden ausführlicher beschrieben.

1. Unterstützung für Kernel 2.6

CS Linux und CS Linux auf zSeries unterstützen jetzt die Serverinstallation unter den Linux-Distributionen mit dem Kernel 2.6 von RedHat und SuSE. Die für die Serverinstallation unterstützten Distributionen sind RHEL4 und SLES9-SP1. Wenn Sie das Produkt CS Linux auf einem Intel-System installieren, muss sich auf diesem System der Kernel einer 32-Bit-Linux-Distribution befinden. Wenn Sie CS Linux auf einem System mit Linux auf POWER (OpenPower oder Power5) installieren, muss sich auf diesem System der Kernel einer 64-Bit-Linux-Distribution befinden. Das Produkt CS Linux auf zSeries kann unter einem 31-Bit- oder 64-Bit-Linux-Kernel installiert werden. Künftig wird es für die zSeries-Plattform jedoch nur noch Linux-Distributionen mit 64-Bit-Kernel geben. Die Anwendungen können 32-Bit- oder 64-Bit-Bibliotheken verwenden.

Client-SNA-Anwendungen, die Remote API Client mit Unterstützung für Linux-Distributionen mit Kernel 2.6 verwenden, können auf Intel-Plattformen nur 32-Bit-Bilbiotheken aufrufen. Wird die Client-Anwendung unter einer x86_64-Linux-Distribution ausgeführt, können daher nur die 32-Bit-Bibliotheken verwendet werden. Auf den Plattformen Linux auf zSeries und Linux auf POWER können Client-SNA-Anwendungen sowohl die 32-Bit- als auch die 64-Bit-Bibliotheken nutzen.

Die folgende Tabelle gibt einen Überblick über die Linux-Kernel-Distributionen, die auf den verschiedenen Plattformen unterstützt werden:

Tabelle 1. Unterstützte Linux-Kernel-Distributionen

Plattform	RHAS2.1	SLES8	RHEL3	SLES9	RHEL4	AIX Version 5
Intel						
Client (i686/32-Bit)	x	x	x	x	x	

Tabelle 1. Unterstützte Linux-Kernel-Distributionen (Forts.)

Plattform	RHAS2.1	SLES8	RHEL3	SLES9	RHEL4	AIX Version 5
Client (x86_64), 32-Bit-API		x	x	x	x	
Server (i686/32-Bit)	x	x	x	x	x	
OpenPower oder Power5 (64-Bit-Kernel, ppc64)						
Client				x	x	
Server				x	x	
Linux auf zSeries (s390/31-Bit)						
Client		x	x	x	x	
Server		x	x	x*	x*	
Linux auf zSeries (s390x, zSeries, 64-Bit)						
Client		x	x	x	x	
Server		x	x	x	x	
AIX						
Client (32-Bit)						x
Client (64-Bit)						x

* kennzeichnet Unterstützung, die bei künftigen Linux-Releases möglicherweise nicht mehr gewährleistet ist.

2. Linux auf POWER

Das Produkt CS Linux (5724-i33) bietet jetzt Unterstützung für mehrere Plattformen, zu denen auch die Plattform Linux auf POWER gehört. Die entsprechenden Server sind OpenPower-Server und die Power5-pSeries-Server. Wenn Sie den Server auf diesen Plattformen installieren möchten, ist RHEL4 oder SLES9-SP1 erforderlich. Falls Sie CS Linux unter Linux auf POWER installieren möchten, lesen Sie die Datei README.ppc64.xx_yy. (Die Angabe xx_yy steht hier für die Ländereinstellung Ihres Systems.)

Remote API Client für Linux auf POWER ist im Lieferumfang der Produkte CS Linux und CS Linux für zSeries enthalten. Die neue Client-Unterstützung ist auf dem Installationsdatenträger im Verzeichnis /ibm-commserver-clients/linux-ppc64 enthalten. Remote API Client wird unter Linux im Benutzerbereich ausgeführt und ist nicht vom Kernel abhängig. Informationen zur Installation des Clients finden Sie unter /ibm-commserver-clients/linux-ppc64/README.

3. Remote API Client für AIX

Communications Server, die in einer Serverdomäne mit Unterstützung für Remote API Client ausgeführt werden, unterstützen jetzt die Ausführung von SNA-Anwendungen auf Plattformen mit AIX Version 5. Wenn Sie Anwendungen für die AIX-Plattform schreiben, sollten Sie die Kompilierung und Links (Tags compile und link) genau wie in Communications Server für AIX beschreiben (siehe http://www.ibm.com/software/network/commserver/library/aix).

Die Verbindung zwischen dem AIX-Client und den Linux-Servern wird über TCP/IP hergestellt. Alle SNA-Anwendungen unter AIX, die die Schnittstellen CPI-C, APPC, LUA und einige NOF-Schnittstellen für Communications Server für AIX ab Version 5 verwenden, können diesen Remote API Client für AIX verwenden. Für die älteren Schnittstellen von CS/AIX Version 4.2 bietet Remote API Client keine Unterstützung.

4. Unterstützung für primäre LU

LUA-Anwendungen stellen eine Verbindung zu Hostgroßrechnern normalerweise als sekundäre LUs her. Die Definition der Sitzungen wird demzufolge von der Hostanwendung gesteuert, die das BIND für den Start einer Sitzung sendet. Communications Server kann jetzt für untergeordnete SNA-abhängige Einheiten über LAN-Schnittstellen mit dem Interface RUI_INIT_PRIMARY als primäre LU agieren. Über dieses Interface kann eine Anwendung ohne einen Hostgroßrechner Sitzungen mit untergeordneten abhängigen LUs aufbauen.

Für die Verwendung primärer LU-Anwendungen muss der Host für untergeordnete LUs (oder eine Schablone für untergeordnete PU) mit dem Host-LU-Namen #PRIRUI# konfiguriert sein. Diese Konfiguration zeigt dem Server an, dass die Anwendungen, die RUI_INIT_PRIMARY nutzen, diese PUs und die ihnen zugeordneten LU-Ressourcen steuern. Die PUs können nur an LAN-Ports verwendet werden. In den folgenden Updates zum LUA Programming Guide finden Sie Informationen zu den Schnittstellen für die Programmierung von Anwendungen, die die Unterstützung für primäre LU nutzen sollen.

5. Updates zu Remote API Clients

5.1 Namensänderung

Für Remote API Client von Communications Server wird jetzt an Stelle des alten Namens ″IBM Remote API Client″ angezeigt.

5.2 Installation von Linux- und AIX-Clients

Falls Sie den Server oder Client von Communications Server auf einem System installieren möchten, auf dem sich bereits eine frühere Version des Codes befindet, müssen Sie den Server oder Client zunächst deinstallieren. Die Konfigurationsdateien werden vom Deinstallations- oder Installationsprozess nicht gelöscht oder modifiziert.

5.3 Windows-Clients

Falls Sie den Communications Server Client auf einem System installieren möchten, auf dem sich bereits eine frühere Version des Codes befindet, müssen Sie die folgenden Schritte ausführen, um Ihre aktuelle Konfiguration zu erhalten:

- Setzen Sie **net stop sxclient** ab, um den Dienst Communications Server Client zu stoppen. Dieser Befehl muss wie hier angegeben abgesetzt werden.
- Schließen Sie das Symbol für den Windows-Client-Monitor. (Dieses Symbol befindet sich in der Regel in der unteren rechten Ecke auf dem Desktop.)
- Installieren Sie das Produkt, ohne die vorherige Version zu entfernen.

Falls Sie die installierte Vorversion des Windows-Clients entfernen, werden die Konfigurationsdaten aus der Windows-Registrierungsdatenbank gelöscht und müssen bei der Installation einer aktuelleren Version erneut eingegeben werden.

IBM Remote API Client für Windows stellt jetzt das neue Diagnose-Tool **snagetpd.exe** für den Kundendienst bereit. Dieses Tool erstellt die komprimierte, selbstentpackende Datei **snapd.exe**. Die Fehlerbestimmungsdatei enthält Folgendes:

- wichtige Inhalte der Registrierungsdatenbank

- den vollständigen Inhalt des Installationsverzeichnisses und aller untergeordneten Verzeichnisse
- Angaben zur Dateiversion aller installierten Binärdateien
- die Position aller Protokoll- und Trace-Dateien
- die Ausgaben der Befehle 'ver', 'ipconfig /all', 'route print', 'netstat -an', 'netstat -a'

Das Tool kopiert alle Protokoll- und Trace-Dateien nach snapd. Wenn Sie vom Kundendienst für Communications Server für Linux aufgefordert werden, Informationen zur Fehlerbestimmung bereitzustellen, senden Sie die Datei snapd.exe. Diese Datei hilft dem Kundendienst bei der Lösung der gemeldeten Probleme.

Remote API Client für Windows enthält folgende APAR-Korrekturen:
- LI70604, LI70605 - Auf einigen Windows-XP-Systemen, bei denen keine der Standardländereinstellungen definiert ist, wird die Installation abgebrochen.
- LI70677, LI70678 - Der Haupt-Thread der Anwendung WinCPIC kann nur ein ALLOCATE absetzen.

6. Updates zum LUA Programmer's Guide

Anmerkung: Da es keine deutsche Fassung dieses Handbuchs gibt, sind die folgenden Informationen englisch.

The following information are additions to the LUA Programmer's Guide for describing the program interface for RUI_INIT_PRIMARY. You should refer to the LUA Programmer's Guide section for RUI_ INIT for any considerations or features that may not be fully described in this section.

6.1 Designing and Writing LUA Applications: SNA Information for RUI Primary

This section contains SNA considerations for writing CS Linux RUI Primary applications for communications with a downstream LU.

This guide does not attempt to explain SNA concepts in detail. If you need specific information about SNA message flows, refer to the documentation for the host application for which you are designing your CS Linux LUA application.

6.2 Responsibilities of the Primary RUI Application

A Primary RUI application has control of both LU-SSCP and PLU-SLU sessions at the Request/Response Unit (RU) level, and can send and receive SNA RUs on these sessions. The PU-SSCP session is internal to CS Linux and the Primary RUI application cannot access it.

Because a Primary RUI application works at the RU level, it has a large degree of control over the data flow to and from the secondary LU. However, it takes greater responsibility than a regular LUA application for ensuring that the SNA messages it sends are valid and that the RU level protocols (for example bracketing and chaining) are used correctly. In particular, note that CS Linux does not attempt to verify the validity of RUs sent by a Primary RUI application.

The Primary RUI application is responsible for:
- Initializing downstream LUs using RUI_INIT_PRIMARY, and terminating them using RUI_TERM
- Processing NOTIFY messages from the secondary LU as secondary applications start and stop
- Processing INIT-SELF and TERM-SELF to activate and deactivate the PLU-SLU session
- Building, sending, receiving and parsing 3270 data stream messages in data RUs
- Implementing RU level protocols (request control, bracketing, chaining, direction)
- Cryptography (if required)
- Compression (if required).

6.3 Restrictions

CS Linux does not support the following for Primary RUI applications:

- Downstream PUs over DLUR
- Dynamically Defined Dependent LUs (DDDLU)
- Sending STSN (to reset sequence numbers, the application should UNBIND and re-BIND the session).

6.4 Configuration Information: RUI_INIT_PRIMARY Link Configuration

For a Primary RUI application communicating with a downstream LU, the only configuration required is the downstream LU (or a Downstream PU template) that as a host LU name of #PRIRUI#.

6.5 RUI_VERBS: RUI_INIT_PRIMARY Verb Control Description

The RUI_INIT_PRIMARY verb establishes the SSCP-LU session for an SNA Primary application that is communicating with a downstream LU. If the RUI application acts as an SNA secondary and communicates with a host LU, it must use RUI_INIT instead of RUI_INIT_PRIMARY.

6.5.1 Supplied Parameters

The application supplies the following parameters (See /opt/ibm/sna/include/lua_c.h):

lua_verb
LUA_VERB_RUI

lua_verb_length
The length in bytes of the LUA verb record. Set this to sizeof(LUA_COMMON).

lua_opcode
LUA_OPCODE_RUI_INIT_PRIMARY

lua_correlator
LUA_OPCODE_RUI_INIT_PRIMARY

lua_luname
The name in ASCII of the LU for which you want to start the session. This must match the name of a downstream LU configured for use with SNA Gateway, or an LU created implicitly from a downstream LU template that as a host LU name of #PRIRUI#.

This parameter must be eight bytes long; pad on the right with spaces, 0x20, if the name is shorter than eight characters.

lua_max_length
The length of a buffer supplied to receive a copy of the ACTLU(+RSP) RU received from the downstream PU. If the application does not need to receive this information, it can specify a null pointer in the lua_data_ptr parameter, in which case it does not need to provide a data buffer.

lua_data_ptr
A pointer to the buffer supplied to receive a copy of the ACTLU(+RSP) RU received from the downstream PU. If the application does not need to receive this information, it can specify a null pointer, and the information will not be returned.

lua_post_handle
A pointer to a callback routine. If the verb completes asynchronously, LUA will call this routine to indicate completion of the verb. For more information, see Designing and Writing LUA Applications.

lua_encr_decr_option

0 Session-level cryptography is not used.

128 Encryption and decryption are performed.

Anmerkung: Encryption and decryption are performed by the application program.

Any other value will result in the return code LUA_ENCR_DECR_LOAD_ERROR.

6.5.2 Returned Parameters

LUA always returns the following parameter:

lua_flag2.async
This flag is set to 1 if the verb completed asynchronously, or 0 if the verb completed synchronously. RUI_INIT_PRIMARY will always complete asynchronously, unless it returns an error such as LUA-_PARAMETER_CHECK.

Other returned parameters depend on whether the verb completed successfully; see the following sections.

Successful Execution

If the verb executes successfully, LUA returns the following parameters.

lua_prim_rc
LUA_OK

lua_sid
A session ID for the new session. This can be used by subsequent verbs to identify this session.

lua_data_length
The length of the ACTLU(+RSP) RU received from the downstream PU. LUA places the data in the buffer specified by lua_data_ptr.

Unsuccessful Execution

If the verb does not complete successfully, LUA returns a primary return code to indicate the type of error and a secondary return code to provide specific details about the reason for the unsuccessful execution.

Please refer to the LUA Programmer's Guide section describing Unsuccessful Execution for RUI_INIT to see the list of possible return codes. In addition to these, the following indications may return for RUI_INIT_PRIMARY specific errors:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT_PRIMARY

An RUI_INIT_PRIMARY verb is currently being processed for this session.

6.5.3 Interaction with Other Verbs

The RUI_INIT_PRIMARY verb must be the first LUA verb issued for the session.

Until this verb has completed successfully, the only other LUA verb that can be issued for this session is RUI_TERM, which will terminate a pending RUI_INIT_PRIMARY.

All other verbs issued on this session must identify the session using one of the following parameters from this verb:

- The session ID, returned to the application in the lua_sid parameter
- The LU name, supplied by the application in the lua_luname parameter

6.5.4 Usage and Restrictions

The RUI_INIT_PRIMARY verb completes after an ACTLU positive response is received from the downstream LU. If necessary, the verb waits indefinitely. If the application needs to check the contents of this ACTLU positive response, it can do so by supplying a data buffer on RUI_INIT_PRIMARY (using the lua_max_length and lua_data_ptr parameters) in which CS Linux returns the contents of the received message.

Once the RUI_INIT_PRIMARY verb has completed successfully, this session uses the LU for which the session was started. No other LUA session (from this or any other application) can use the LU until the RUI_TERM verb is issued, or until an LUA_SESSION_FAILURE primary return code is received.

If the RUI_INIT_PRIMARY verb returns with an LUA_IN_PROGRESS primary return code then the Session ID will be returned in the lua_sid parameter. This Session ID is the same as that returned when the verb completes successfully and can be used with the RUI_TERM verb to terminate an outstanding RUI-_INIT_PRIMARY verb.

---

7. Updates zum Administratorhandbuch

Für das Verwaltungs-Tool snaadmin gibt es die beiden folgenden neuen Befehle:

define_rtp_tuning
Definiert neue Zeitgeber für die Optimierung der Konnektivität von HPR-Sitzungen

query_rtp_tuning
Fragt HPR-Zeitgeber für Pfadumschaltung ab

Wenn Sie diese Befehle verwenden möchten, können Sie den Hilfebefehl ″snaadmin -d -h define_rtp_tuning″ oder ″snaadmin -d -h query_rtp_tuning″ absetzen, um die Syntax der Befehle anzuzeigen. Detaillierte Informationen hierzu finden Sie auch in den Updates zum NOF Programmer's Guide.

---

8. Updates zum NOF Programmer's Guide

Anmerkung: Da es keine deutsche Fassung dieses Handbuchs gibt, sind die folgenden Informationen englisch.

8.1 DEFINE_RTP_TUNING

DEFINE_RTP_TUNING specifies parameters to be used when setting up RTP connections. After you issue this verb, the parameters you specify will be used for all future RTP connections until you modify them by issuing a new DEFINE_RTP_TUNING verb.

8.1.1 Supplied Parameters

The application supplies the following parameters (See define_rtp_tuning structure in /opt/ibm/sna/include/nof_c.h):

opcode
AP_DEFINE_RTP_TUNING

path_switch_attempts
Number of path switch attempts to set on new RTP connections. Specify a value in the range 1-255. If you specify 0(zero), CS Linux uses the default value of 6.

short_req_retry_limit
Number of times a Status Request is sent before CS Linux determines that an RTP connection is disconnected and starts Path Switch processing. Specify a value in the range 1-255. If you specify 0(zero), CS Linux uses the default value of 6.

path_switch_times
Length of time in seconds for which CS Linux attempts to path switch a disconnected RTP connection. This parameter is specified as four separate time limits for each of the valid transmission priorities in order: AP_LOW, AP_MEDIUM , AP_HIGH, and AP_NETWORK. Each of these must be in the range 1-65535. The value you specify for each transmission priority must not exceed the value for any lower transmission priority.

If you specify 0(zero) for any of these values, CS Linux uses the corresponding default value as follows:

- 480 seconds (8 minutes) for AP_LOW
- 240 seconds (4 minutes) for AP_MEDIUM
- 120 seconds (2 minutes) for AP_HIGH
- 60 seconds (1 minute) for AP_NETWORK

8.1.2 Returned Parameters

Successful Execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

Unsuccessful Execution

If the verb does not execute because of a parameter error, CS Linux returns the following parameters:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
Folgende Werte sind gültig:

AP_INVALID_PATH_SWITCH_TIMES The path_switch_times parameter was not valid; for example, you may have specified a value for one transmission priority that exceeds the value specified for a lower transmission priority.

Common Return Codes lists further secondary return codes associated with AP_PARAMETER-_CHECK, which are common to all NOF verbs.

8.2 QUERY_RTP_TUNING

QUERY_RTP_TUNING returns information about the parameters that will be used for future RTP connections. This information was previously set up using DEFINE_RTP_TUNING.

8.2.1 Supplied Parameters

The application supplies the following parameters (See query_rtp_tuning structure in /opt/ibm/sna/include/nof_c.h):

opcode
AP_QUERY_RTP_TUNING

8.2.2 Returned Parameters

Successful Execution

If the verb executes successfully, CS Linux returns the following parameters:

primary_rc
AP_OK

path_switch_attempts
Number of path switch attempts to set on new RTP connections

short_req_retry_limit
Number of times a Status Request is sent before CS Linux determines that an RTP connection is disconnected and starts Path Switch processing.

path_switch_times
Length of time in seconds for which CS Linux attempts to path switch a disconnected RTP connection. This parameter is specified as four separate time limits for each of the valid transmission priorities in order: AP_LOW, AP_MEDIUM , AP_HIGH, and AP_NETWORK.

Other Conditions

Common Return Codes lists further combinations of primary and secondary return codes that are common to all NOF verbs.