Linux용 IBM 통신 서버 버전 6.2.1의 새로운 기능

이 문서는 Linux용 통신 서버의 버전 6.2.1.0(제품 번호: 5724-i33, CS Linux) 및 zSeries의 Linux용 통신 서버(제품 번호: 5724-i34, zSeries의 CS Linux)에 대한 유지보수 릴리스 변경사항을 설명합니다. 설명은 Linux용 통신 서버 및 zSeries의 Linux용 통신 서버 제품 모두에 적용됩니다.

이 Linux용 통신 서버 버전 6.2.1.0의 유지보수 릴리스에 제공된 변경사항은 다음을 통합합니다.

1. 2.6 커널 지원 - 갱신된 Linux 운영 체제 지원
2. Linux on Power - 추가 플랫폼 지원 및 패키지
3. AIX 원격 API 클라이언트 - 추가 플랫폼 및 패키지
4. 1차 LU 지원 - 1차 LU 0 흐름을 지원하는 LUA 프로그래밍 인터페이스
5. 원격 API 클라이언트 갱신사항 - 클라이언트에 제공된 갱신사항
6. LUA Programmer's Guide 갱신사항 - 추가 1차 LU 인터페이스
7. Administration Guide updates - Additional commands to define and query HPR timer parameters (이 안내서는 번역되지 않았습니다. 영문으로 제공됩니다.)
8. NOF Programmer's Guide updates - Additional verbs to define and query HPR timer parameters (이 안내서는 번역되지 않았습니다. 영문으로 제공됩니다.)

이 새로운 기능은 다음에서 자세히 설명됩니다.

1. 2.6 커널 지원

CS Linux 및 zSeries의 CS Linux는 Red Hat 및 SuSE에서 2.6 Linux 커널 분배에 대해 서버 설치를 지원합니다. 서버 설치에 지원되는 분배는 RHEL 4 및 SLES 9-SP1입니다. Intel 시스템에 CS Linux 제품을 설치할 때 커널은 32비트 Linux 분배여야 합니다. Linux on Power(OpenPower 또는 Power5) 시스템에 CS Linux를 설치할 때 커널은 64비트 Linux 분배여야 합니다. CS Linux on zSeries 제품은 31비트 또는 64비트 Linux 커널에 설치되어야 합니다. 그러나 앞으로는 zSeries 플랫폼용 Linux 분배는 64비트 커널 전용으로만 지원됩니다. 응용프로그램은 32비트 또는 64비트 라이브러리를 사용할 수 있습니다.

2.6 Linux 커널 분배 지원이 있는 원격 API 클라이언트의 경우 클라이언트 SNA 응용프로그램은 Intel 플랫폼에서 32비트 라이브러리만 호출할 수 있습니다. 따라서 클라이언트 응용프로그램이 x86_64 Linux 분배에서 실행 중일 경우 32비트 라이브러리만 사용될 수 있습니다. Linux on zSeries 및 Linux on Power 플랫폼의 경우 클라이언트 SNA 응용프로그램은 32비트 또는 64비트 라이브러리를 사용할 수 있습니다.

여러 플랫폼에서 지원되는 Linux 커널 분배는 다음의 테이블에서 맵핑됩니다.

표 1. 지원되는 Linux 커널 분배

플랫폼	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX 버전5
Intel						

표 1. 지원되는 Linux 커널 분배 (계속)

플랫폼	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX 버전5
클라이언트(i686 - 32비트)	x	x	x	x	x	
클라이언트 (x86_64), 32비트 API		x	x	x	x	
서버(i686 - 32비트)	x	x	x	x	x	
OpenPower* 또는 *Power 5(64비트 커널, ppc64)						
클라이언트				x	x	
서버				x	x	
Linux on zSeries(s390, 31비트)						
클라이언트		x	x	x	x	
서버		x	x	x*	x*	
Linux on zSeries(s390x, zSeries, 64비트)						
클라이언트		x	x	x	x	
서버		x	x	x	x	
AIX						
클라이언트 - 32비트						x
클라이언트 - 64비트						x

별표(*)는 향후 Linux 릴리스에서 사용할 수 없는 지원을 표시합니다.

2. Linux on Power

CS Linux 제품(5724-i33)은 OpenPower 서버 및 Power5 pSeries 서버인 Linux on Power 플랫폼용 CS Linux 서버를 포함하는 다중 플랫폼을 지원합니다. 서버를 설치할 때 이 플랫폼을 지원하려면 RHEL 4 또는 SLES 9-SP1이 필요합니다. Linux on Power에 CS Linux 서버를 설치하려면 README.ppc64.xx_yy를 참조하십시오(여기서 xx_yy는 시스템의 로케일임).

Linux on Power용 CS Linux 원격 API 클라이언트는 CS Linux 및 CS Linux for zSeries 제품과 함께 제공됩니다. 새 클라이언트 지원은 설치 디스크의 /ibm-commserver-clients/linux-ppc64 디렉토리에서 찾을 수 있습니다. 원격 API 클라이언트는 Linux의 "사용자 영역"에서 실행되며 커널 종속이 아닙니다. 클라이언트를 설치하려면 /ibm-commserver-clients/linux-ppc64/README를 참조하십시오.

3. AIX 원격 API 클라이언트

원격 API 클라이언트를 지원하는 서버의 도메인에서 실행되는 통신 서버는 AIX 버전5 플랫폼에서 실행되는 SNA 응용프로그램을 지원할 수 있습니다. AIX 플랫폼용 응용프로그램을 작성할 때 AIX용 통신 서버에 있는 컴파일 및 링크에 동일한 설명을 사용해야 합니다(http://www.ibm.com/software/network/commserver/library/aix).

AIX 클라이언트 및 Linux 서버 간의 연결은 TCP/IP를 통해 수행됩니다. AIX용 통신 서버 버전5 이상에 대해 CPI-C, APPC, LUA 및 일부 NOF 인터페이스를 사용하는 AIX용 SNA 응용프로그램은 이 AIX 원격 API 클라이언트를 사용할 수 있습니다. 이전 CS/AIX V4.2 인터페이스는 원격 API 클라이언트에서 지원되지 않습니다.

4. 1차 LU 지원

LUA 응용프로그램은 보통 2차 LU로 호스트 메인프레임에 연결됩니다. 이는 세션 정의가 세션을 시작하기 위해 BIND를 송신하는 호스트 응용프로그램을 통해 제어됨을 의미합니다. 통신 서버에는 RUI_INIT_PRIMARY 인터페이스를 사용하여 LAN 인터페이스를 통해 다운스트림 SNA 종속 장치에 대한 1차 LU로 작동하는 기능이 있습니다. 이 인터페이스를 사용하면 응용프로그램은 호스트 메인프레임 없이 다운스트림 종속 LU 세션에 연결할 수 있습니다.

1차 LU 응용프로그램을 사용하려면 #PRIRUI#의 호스트 LU 이름이 있는 다운스트림 LU(또는 다운스트림 PU 템플리트)와 함께 노드를 구성해야 합니다. 이는 RUI_INIT_PRIMARY를 사용하는 응용프로그램이 이 응용프로그램에 지정된 PU 및 LU 자원을 제어함을 서버에 표시합니다. PU는 LAN 포트에서만 사용될 수 있습니다. 1차 LU 지원을 사용할 응용프로그램을 프로그래밍하는 인터페이스 정보는 LUA Programming Guide 갱신사항(아래)을 참조하십시오.

5. 원격 API 클라이언트 갱신사항

5.1 이름 변경

통신 서버 원격 API 클라이언트는 이전 이름 대신 ″IBM 원격 API 클라이언트″ 이름을 표시합니다.

5.2 Linux 및 AIX 클라이언트 설치

이미 이전 코드 버전이 있는 시스템에 통신 서버의 서버 또는 클라이언트를 설치할 경우에 먼저 서버 또는 클라이언트를 설치 제거해야 합니다. 구성 파일은 설치 제거 및 설치 프로세스로 삭제되거나 수정되지 않습니다.

5.3 Windows 클라이언트

이미 이전 코드 버전이 있는 시스템에 통신 서버 클라이언트를 설치할 경우 현재 구성을 유지하려면 다음 단계를 수행해야 합니다.

- **net stop sxclient**를 실행하여 통신 서버 클라이언트 서비스를 중지하십시오. 이 명령을 표시된 대로 입력해야 합니다(변환하지 말 것).

- Windows 클라이언트 모니터 아이콘(보통 데스크탑의 오른쪽 하단 부분에 있음)을 닫으십시오.
- 이전 버전을 제거하지 않고 제품을 설치하십시오.

설치되어 있는 Windows 클라이언트의 이전 버전을 제거할 경우 구성 정보가 Windows 레지스트리 데이터베이스에서 삭제되므로 최신 버전을 설치하기 전에 구성 정보를 다시 입력해야 합니다.

Windows용 IBM 원격 API 클라이언트에는 서비스 진단 도구인 **snagetpd.exe**가 포함되어 있습니다. 이 도구는 자체 추출하는 압축 파일인 **snapd.exe**를 작성합니다. 문제점 판별 파일에는 다음이 포함되어 있습니다.

- 관련 레지스트리 내용
- 설치 디렉토리 및 설치 디렉토리 아래에 있는 모든 디렉토리 내용
- 설치된 모든 2진에 대한 파일 버전 정보
- 모든 로그 및 추적 파일의 위치
- 'ver', 'ipconfig /all', 'route print', 'netstat -an', 'netstat -a' 명령의 출력

도구는 모든 로그 및 추적 파일을 snapd에 복사합니다. 통신 서버 Linux 서비스가 문제점 판별 정보를 제공하도록 지시하면 snapd.exe 파일을 송신해야 합니다. 이 파일은 보고된 문제점에 대한 서비스를 제공하는 데 도움이 됩니다.

Windows 원격 API 클라이언트에 포함된 APAR 수정사항은 다음과 같습니다.

- LI70604, LI70605 - 디폴트 로케일 중 하나가 설정되어있지 않은 일부 Windows XP 시스템의 경우 설치가 중단됩니다.
- LI70677, LI70678 - WinCPIC 응용프로그램의 기본 스레드는 둘 이상의 ALLOCATE를 실행할 수 없습니다.

---

6. LUA Programmer's Guide Updates(이 안내서는 번역되지 않았습니다. 영문으로 제공됩니다.)

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.

주: 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. 관리 안내서 갱신사항

snaadmin 관리 도구의 다음과 같은경우 2개의 새로운 명령이 있습니다.

define_rtp_tuning

HPR 세션 연결성을 조정하는 새 타이머를 정의합니다.

query_rtp_tuning

HPR 경로 전환 타이머를 조회합니다.

이 명령을 사용하려면 도움말 명령인 ″snaadmin -d -h define_rtp_tuning″ 또는 ″snaadmin -d -h query_rtp_tuning″을 사용하여 명령 구문을 확인하십시오. 자세한 정보는 NOF Programmer's Guide 갱신사항을 참조할 수도 있습니다.

---

8. NOF Programmer's Guide Updates(이 안내서는 번역되지 않았습니다. 영문으로 제공됩니다.)

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

Possible values are:

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.