IBM Communications Server for Linux V6.2.1 中的新增内容

本文档描述了版本 6.2.1.0 的 Communications Server for Linux（产品号 5724-i33，CS Linux）和 Communications Server for Linux on zSeries（产品号 5724-i34，CS Linux on zSeries）的维护版更改。其中的描述适用于 Communications Server for Linux 和 Communications Server for Linux on zSeries 产品。

Communications Server for Linux V6.2.1.0 的本维护版中提供的更改合并了以下内容:

1. 2.6 内核支持 – 更新的 Linux 操作系统支持
2. Linux on Power – 其它平台支持和打包方法
3. AIX Remote API Client – 其它平台和打包方法
4. 主 LU 支持 – 支持主 LU 0 流的 LUA 编程接口
5. Remote API Client 更新 – 为客户机提供的更新
6. LUA Programmer's Guide 更新 – 其它主 LU 接口
7. Administration Guide 更新 – 用于定义和查询 HPR 计时器参数的其它命令
8. NOF Programmer's Guide 更新 – 用于定义和查询 HPR 计时器参数的其它动词

下面更详细地描述了此新功能。

1. 2.6 内核支持

CS Linux 和 CS Linux on zSeries 现在支持从 Red Hat 和 SuSE 的 2.6 Linux 内核分发版上进行服务器安装。服务器安装所支持的分发版是 RHEL 4 和 SLES 9-SP1。当在 Intel 系统上安装 CS Linux 产品时，内核必须是 32 位 Linux 分发版。当在 Linux on Power（OpenPower 或 Power5）系统上安装 CS Linux 时，内核必须是 64 位 Linux 分发版。CS Linux on zSeries 产品可以安装在 31 位或 64 位 Linux 内核上。但以后 zSeries 平台的 Linux 分发版将仅适用于 64 位内核。应用程序可以使用 32 位或 64 位库。

对于已具有 2.6 Linux 内核分发版支持的 Remote API Client，客户机 SNA 应用程序只能调用 Intel 平台上的 32 位库。因此，如果客户机应用程序在 x86_64 Linux 分发版上运行，则只能使用 32 位库。对于 Linux on zSeries 和 Linux on Power 平台，客户机 SNA 应用程序可以使用 32 位或 64 位库。

下表中列出了在各种平台上受支持的 Linux 内核分发版的对应关系:

表 1. 受支持的 Linux 内核分发版

平台	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
Intel						
客户机（i686 – 32 位）	x	x	x	x	x	
客户机（x86_64），32 位 API		x	x	x	x	
服务器（i686 – 32 位）	x	x	x	x	x	

表 1. 受支持的 Linux 内核分发版（续）

平台	RHAS 2.1	SLES 8	RHEL 3	SLES 9	RHEL 4	AIX V5
OpenPower 或 Power 5（64 位内核，ppc64）						
客户机				x	x	
服务器				x	x	
Linux on zSeries（s390，31 位）						
客户机		x	x	x	x	
服务器		x	x	x*	x*	
Linux on zSeries（s390x 和 zSeries，64 位）						
客户机		x	x	x	x	
服务器		x	x	x	x	
AIX						
客户机 - 32 位						x
客户机 - 64 位						x

* 指示在将来的 Linux 发行版中可能不推荐的支持

2. Linux on Power

CS Linux 产品（5724-i33）现在支持多平台，包括支持 Linux on Power 平台上的 CS Linux 服务器，这些服务器是 OpenPower 服务器和 Power5 pSeries 服务器。当安装服务器时，这些平台的支持需要 RHEL 4 或 SLES 9-SP1。要在 Linux on Power 上安装 CS Linux 服务器，请参阅 README.ppc64.xx_yy（其中 xx_yy 是系统的语言环境）。

CS Linux Remote API Client for Linux on Power 是与 zSeries 和 CS Linux for zSeries 产品一起提供的。可以在安装磁盘的 /ibm-commserver-clients/linux-ppc64 目录中找到新的客户机支持。Remote API Client 在 Linux 上的“用户空间”中运行且与内核无关。要安装该客户机，请参阅 /ibm-commserver-clients/linux-ppc64/README。

3. AIX Remote API Client

在支持 Remote API Client 的服务器的域中运行的 Communications Server 现在可以为在 AIX V5 平台上运行的 SNA 应用程序提供支持。当编写 AIX 平台的应用程序时，您应使用在 Communications Server for AIX（http://www.ibm.com/software/network/commserver/library/aix）中找到的相同编译和链接描述。

AIX 客户机和 Linux 服务器之间的连接将基于 TCP/IP。AIX 上将 CPI-C、APPC、LUA 和某些 NOF 接口用于 Communications Server for AIX V5 和更高版本的 SNA 应用程序可以使用此 AIX Remote API Client。Remote API Client 不支持较旧的 CS/AIX V4.2 接口。

4. 主 LU 支持

LUA 应用程序通常作为辅助 LU 连接至主机。这意味着会话定义由发送 BIND 以启动会话的主机应用程序控制。Communications Server 现在能够基于使用 RUI_INIT_PRIMARY 接口的 LAN 接口充当下游 SNA 从属设备的主 LU。通过使用此接口，应用程序可以连接下游从属 LU 会话而不需要主机。

要使用主 LU 应用程序，必须用主机 LU 名为 #PRIRUI# 的下游 LU（或下游 PU 模板）来配置节点。这将对服务器表明，使用 RUI_INIT_PRIMARY 的应用程序将控制这些 PU 和分配给这些 PU 的 LU 资源。只能在 LAN 端口上使用这些 PU。有关编写要使用主 LU 支持的应用程序的接口信息，请参阅 LUA Programming Guide 更新（下面）。

5. Remote API Client 更新

5.1 名称更改

Communications Server Remote API Client 现在将显示名称 “IBM Remote API Client” 来代替先前的名称。

5.2 Linux 和 AIX 客户机安装

如果您要在已具有先前版本代码的系统上安装 Communications Server 服务器或客户机，则必须首先卸载服务器或客户机。卸载和安装过程将不会删除或修改配置文件。

5.3 Windows 客户机

如果您要在已具有先前版本代码的系统上安装 Communications Server 客户机，则必须执行下列步骤以保留当前配置:

- 通过发出 **net stop sxclient** 命令停止 Communications Server 客户机服务。必须按看到的内容（即未翻译的）输入此命令。
- 关闭 Windows 客户机监视器图标（通常可在桌面的右下部找到）。
- 安装该产品而不必除去先前版本。

如果除去已安装的先前 Windows 客户机，则会从 Windows 注册数据库中删除配置信息，因此在安装更高版本之后必须再次输入配置信息。

IBM Remote API Client for Windows 现在包括服务诊断工具 **snagetpd.exe**。此工具创建一个压缩文件 **snapd.exe**，该文件是自解压的。问题确定文件包含以下内容:

- 有关的注册表内容
- 安装目录中或其下的一切内容的目录内容
- 所有已安装的二进制文件的文件版本信息
- 所有日志和跟踪文件的位置
- 命令 “ver”、“ipconfig /all”、“route print”、“netstat -an”、“netstat -a” 的输出

该工具将所有日志和跟踪文件复制到 snapd。当 Communications Server Linux 服务指示提供问题确定信息时，您应发送 snapd.exe 文件。该文件将有助于为报告的问题提供服务。

Windows Remote API Client 包括的 APAR 修订如下:

- LI70604 和 LI70605 – 在某些未设置一个缺省语言环境的 Windows XP 系统上安装异常中断。
- LI70677 和 LI70678 – WinCPIC 应用程序的主线程不能发出多个 ALLOCATE。

6. LUA Programmer's Guide 更新

下列信息是对 LUA Programmer's Guide 的补充，用于描述 RUI_INIT_PRIMARY 程序接口。有关 RUI_INIT 的内容，您应参阅 LUA Programmer's Guide 中的章节以了解可能未在本节中详尽描述的任何注意事项或功能。

6.1 设计和编写 LUA 应用程序: 主 RUI 的 SNA 信息

本节包含有关编写 CS Linux 主 RUI 应用程序以便与下游 LU 通信的 SNA 注意事项。

本指南并不尝试详细说明 SNA 概念。如果您需要关于 SNA 消息流的特定信息，请参阅您要为其设计 CS Linux LUA 应用程序的主机应用程序的文档。

6.2 主 RUI 应用程序的职责

主 RUI 应用程序可以在请求／响应单元（RU）级别控制 LU-SSCP 和 PLU-SLU 会话，并且可以在这些会话上发送和接收 SNA RU。PU-SSCP 会话是 CS Linux 的内部会话，主 RUI 应用程序不能访问它。

因为主 RUI 应用程序在 RU 级别工作，所以它对进出辅助 LU 的数据流有很大程度的控制。但是在确保它所发送的 SNA 消息有效以及正确地使用 RU 级别协议（例如，用括号分组和组链）方面，它所担负的责任要比一般 LUA 应用程序更重。特别注意，CS Linux 不会尝试验证由主 RUI 应用程序发送的 RU 的有效性。

主 RUI 应用程序的职责如下:

- 使用 RUI_INIT_PRIMARY 初始化下游 LU，并使用 using RUI_TERM 终止它们
- 当辅助应用程序启动和停止时处理来自辅助 LU 的通知消息
- 处理 INIT-SELF 和 TERM-SELF 以激活和取消激活 PLU-SLU 会话
- 构建、发送、接收和解析数据 RU 中的 3270 数据流消息
- 实现 RU 级别协议（请求控制、编组、组链和方向）
- 密码术（如果需要的话）
- 压缩（如果需要的话）

6.3 限制

CS Linux 不支持主 RUI 应用程序的以下各项:

- 基于 DLUR 的下游 PU
- 动态定义的从属 LU（DDDLU）
- 发送 STSN（以复位序号，应用程序应取消绑定并重新绑定会话）。

6.4 配置信息: RUI_INIT_PRIMARY 链接配置

对于主 LU 应用程序，每个应用程序控制的 LU 必须配置为带有名为 #PRIRUI# 的主机 LU 的下游 LU（或下游 PU 模板）。

6.5 RUI_VERBS: RUI_INIT_PRIMARY 动词控制描述

RUI_INIT_PRIMARY 动词为与下游 LU 通信的主 SNA 应用程序建立 SSCP-LU 会话。如果 RUI 应用程序充当辅助 SNA 并与主机 LU 通信，则它必须使用 RUI_INIT 代替 RUI_INIT_PRIMARY。

6.5.1 提供的参数

该应用程序提供下列参数（请参阅 /opt/ibm/sna/include/lua_c.h）:

lua_verb
LUA_VERB_RUI

lua_verb_length
LUA 动词记录的长度（以字节计）。将此参数设置为 sizeof(LUA_COMMON)。

lua_opcode
LUA_OPCODE_RUI_INIT_PRIMARY

lua_correlator
LUA_OPCODE_RUI_INIT_PRIMARY

lua_luname
您要启动会话的 LU 的 ASCII 名称。此名称必须与为了供 SNA 网关使用而配置的下游 LU 的名称匹配，或与根据主机 LU 名为 #PRIRUI# 的下游 LU 模板而隐式创建的 LU 的名称匹配。

此参数的长度必须为 8 个字节 如果名称的长度少于 8 个字符，则在右边填充空格 0x20。

lua_max_length
提供的缓冲区的长度，该缓冲区用来接收从下游 PU 接收到的 ACTLU(+RSP) RU 的副本。如果应用程序不需要接收此信息，则它可以在 lua_data_ptr 参数中指定一个空指针，在这种情况下，它不需要提供数据缓冲区。

lua_data_ptr
提供的缓冲区的指针，该缓冲区用来接收从下游 PU 接收到的 ACTLU(+RSP) RU 的副本。如果应用程序不需要接收此信息，则它可以指定一个空指针，并且此信息将不会返回。

lua_post_handle
回调例程的指针。如果动词以异步方式完成，则 LUA 将调用此例程来指示动词是否已完成。有关更多信息，请参阅『设计和编写 LUA 应用程序』。

lua_encr_decr_option

0 不使用会话级别密码术。

128 执行加密和解密。

注: 由应用程序执行加密和解密。

其它任何值都将产生返回码 LUA_ENCR_DECR_LOAD_ERROR。

6.5.2 返回的参数

LUA 始终返回下列参数:

lua_flag2.async
如果动词以异步方式完成，则此标志设置为 1，如果动词以同步方式完成，则此标志设置为 0。除非 RUI_INIT_PRIMARY 返回诸如 LUA_PARAMETER_CHECK 等错误，否则它将始终以异步方式完成。

其它返回参数取决于动词是否成功完成，请参阅以下各节。

成功执行

如果动词成功执行，则 LUA 返回下列参数。

lua_prim_rc
LUA_OK

lua_sid
新会话的会话标识。后续动词可以使用它来标识此会话。

lua_data_length
从下游 PU 接收到的 ACTLU(+RSP) RU 的长度。LUA 将数据放置在由 lua_data_ptr 指定的缓冲区中。

未成功的执行

如果动词未成功完成，则 LUA 返回主返回码以指示错误类型，并返回辅助返回码以提供关于未成功执行原因的特定详细信息。

请参阅 LUA Programmer's Guide 中描述 RUI_INIT 未成功执行的章节以查看可能的返回码列表。除了这些以外，还可能返回以下指示来表示 RUI_INIT_PRIMARY 特定错误:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
LUA_DUPLICATE_RUI_INIT_PRIMARY

正在为此会话处理 RUI_INIT_PRIMARY 动词。

6.5.3 与其它动词的交互

RUI_INIT_PRIMARY 动词必须是对该会话发出的第一个 LUA 动词。

在此动词已成功完成前，唯一可以对此会话发出的其它 LUA 动词是 RUI_TERM，它将终止暂挂 RUI_INIT_PRIMARY。

在此会话上发出的所有其它动词必须使用来自此动词的以下其中一个参数来标识会话:

- lua_sid 参数中返回至应用程序的会话标识
- lua_luname 参数中由应用程序提供的 LU 名

6.5.4 使用和限制

RUI_INIT_PRIMARY 动词在从下游 LU 接收到 ACTLU 肯定响应之后完成。必要时，该动词会无限期地等待。如果应用程序需要检查此 ACTLU 肯定响应的内容，则它可以通过在 RUI_INIT_PRIMARY 上提供数据缓冲区来达到此目的（使用 lua_max_length 和 lua_data_ptr 参数），这样，CS Linux 就会返回接收到的消息的内容。

一旦 RUI_INIT_PRIMARY 动词成功完成，此会话就会使用启动该会话的 LU。在发出 RUI_TERM 动词之后，或在接收到 LUA_SESSION_FAILURE 主返回码之后，其它 LUA 会话（来自此应用程序或任何其它应用程序）才能使用该 LU。

如果 RUI_INIT_PRIMARY 动词返回 LUA_IN_PROGRESS 主返回码，则会话标识将在 lua_sid 参数中返回。此会话标识与在动词成功完成时返回的会话标识相同，它可以与 RUI_TERM 动词配合使用以终止未完成的 RUI_INIT_PRIMARY 动词。

7.《管理指南》更新

snaadmin 管理工具增加了 2 个新的命令。这些命令如下:

define_rtp_tuning
定义新的计时器以调整 HPR 会话连接。

query_rtp_tuning
查询 HPR 路径切换计时器

要使用这些命令，发出帮助命令 “snaadmin -d -h define_rtp_tuning” 或 “snaadmin -d -h query_rtp_tuning” 来查看这些命令的语法。您还可以参阅 NOF Programmer’s Guide 更新以了解更具体的信息。

8. NOF Programmer’s Guide 更新

8.1 DEFINE_RTP_TUNING

DEFINE_RTP_TUNING 指定当建立 RTP 连接时要使用的参数。在发出此动词之后，您指定的参数将用于以后的所有 RTP 连接，直到您通过发出新 DEFINE_RTP_TUNING 动词来修改它们为止。

8.1.1 提供的参数

该应用程序提供下列参数（请参阅 /opt/ibm/sna/include/nof_c.h 中的 define_rtp_tuning 结构）：

opcode
AP_DEFINE_RTP_TUNING

path_switch_attempts
要在新的 RTP 连接上设置的路径切换尝试次数。指定一个范围 1 到 255 之间的值。如果您指定 0（零），则 CS Linux 使用缺省值 6。

short_req_retry_limit
在 CS Linux 确定 RTP 连接已断开并启动 “路径切换” 处理之前发送的 “状态请求” 的次数。指定一个范围 1 到 255 之间的值。如果您指定 0（零），则 CS Linux 使用缺省值 6。

path_switch_times
CS Linux 尝试对断开的 RTP 连接进行路径切换的时间长度，以秒计。将以四个单独的时间限制的形式对每个有效的传输优先级（顺序为 AP_LOW、AP_MEDIUM、AP_HIGH 和 AP_NETWORK）指定此参数。其中每个值都必须在 1 到 65535 之间。您对每个传输优先级指定的值都不得超过任何更低传输优先级的值。

如果您对其中任何值指定 0（零），则 CS Linux 使用如下的相应缺省值:

- 对于 AP_LOW，缺省值为 480 秒（8 分钟）
- 对于 AP_MEDIUM，缺省值为 240 秒（4 分钟）
- 对于 AP_HIGH，缺省值为 120 秒（2 分钟）
- 对于 AP_NETWORK，缺省值为 60 秒（1 分钟）

8.1.2 返回的参数

成功执行

如果动词成功执行，则 CS Linux 返回下列参数。

primary_rc
AP_OK

未成功执行

如果由于参数错误导致动词未执行，则 CS Linux 返回以下参数:

primary_rc
AP_PARAMETER_CHECK

secondary_rc
可能的值为:

AP_INVALID_PATH_SWITCH_TIMES path_switch_times 参数无效，例如，您可能为一个传输优先级指定的值超过了为更低传输优先级指定的值。

公共返回码列示了与 AP_PARAMETER_CHECK 相关联的进一步的辅助返回码，这些返回码是所有 NOF 动词的公共返回码。

8.2 QUERY_RTP_TUNING

QUERY_RTP_TUNING 返回关于将用于将来 RTP 连接的参数的信息。此信息先前是使用 DEFINE_RTP_TUNING 设置的。

8.2.1 提供的参数

该应用程序提供下列参数（请参阅 /opt/ibm/sna/include/nof_c.h 中的 query_rtp_tuning 结构）：

opcode
AP_QUERY_RTP_TUNING

8.2.2 返回的参数

成功执行

如果动词成功执行，则 CS Linux 返回下列参数。

primary_rc
AP_OK

path_switch_attempts
要在新的 RTP 连接上设置的路径切换尝试次数

short_req_retry_limit
在 CS Linux 确定 RTP 连接已断开并启动“路径切换”处理之前发送的“状态请求”的次数。

path_switch_times
CS Linux 尝试对断开的 RTP 连接进行路径切换的时间长度，以秒计。将以四个单独的时间限制的形式对每个有效的传输优先级（顺序为 AP_LOW、AP_MEDIUM、AP_HIGH 和 AP_NETWORK）指定此参数。

其它情况

公共返回码列示了所有 NOF 动词的公共主返回码和辅助返回码的进一步组合。