

IBM Communications Server para Linux



Guía del programador para LUA

Versión 6.2.2

IBM Communications Server para Linux



Guía del programador para LUA

Versión 6.2.2

Nota:

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general que figura en el Apéndice B, "Avisos", en la página 165.

Tercera edición (julio de 2006)

Esta publicación es la traducción del original inglés *IBM Communications Server for Linux, LUA Programmer's Guide*, (SC31-6776-01).

Esta edición se aplica a IBM Communications Server para Linux, Versión 6.2.2, y a todos los releases y las modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones o boletines técnicos.

Puede solicitar publicaciones a través del representante local o de la sucursal local de IBM. No hay existencias de publicaciones en la dirección indicada más abajo.

IBM agradece sus comentarios. Al final de esta publicación encontrará una hoja de comentarios del lector. Si dicha hoja no existe, puede enviar sus comentarios a la dirección siguiente:

IBM España, S.A.
National Language Solutions Center
Avda. Diagonal 571
"Edif. L'Illa"
08029 Barcelona
España

Si prefiere enviar sus comentarios de forma electrónica, utilice uno de los métodos siguientes:

- Internet: HOJACOM@es.ibm.com
- Fax: (34)-93-321-6134

Cuando se envía información a IBM, se otorga a IBM un derecho no exclusivo de utilizar o distribuir la información del modo que considere oportuno, sin incurrir por ello en ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1998, 2006. Reservados todos los derechos.

Contenido

| | |
|--|------------|
| Tablas | vii |
| Figuras | ix |
| Acerca de este manual | xi |
| A quién va dirigido este manual | xi |
| Cómo utilizar este manual | xii |
| Organización de este manual | xii |
| Convenios tipográficos | xii |
| Convenios gráficos | xiii |
| Publicaciones relacionadas | xiii |
| Capítulo 1. Conceptos. | 1 |
| ¿Qué es LUA? | 1 |
| Elección de la interfaz a utilizar | 2 |
| LU y sesiones | 2 |
| Configuración | 4 |
| Verbos LUA | 5 |
| Resumen de verbos RUI | 5 |
| Resumen de verbos SLI | 6 |
| Finalización asíncrona de verbos. | 7 |
| Ejemplo de secuencia de comunicaciones LUA | 8 |
| Compatibilidad con LUA | 12 |
| Capítulo 2. Diseño y desarrollo de aplicaciones LUA. | 13 |
| Puntos de entrada LUA para aplicaciones AIX o Linux | 13 |
| Llamada de función RUI | 14 |
| Llamada de función SLI | 14 |
| Parámetros proporcionados | 14 |
| Valores devueltos | 14 |
| Uso | 15 |
| Rutina de devolución de llamada para la finalización asíncrona de verbos | 16 |
| Puntos de entrada LUA para aplicaciones Windows | 16 |
| RUI | 18 |
| WinRUIStartup | 19 |
| WinRUI | 21 |
| WinRUIGetLastInItStatus | 23 |
| WinRUICleanup | 26 |
| GetLuaReturnCode | 27 |
| SLI | 27 |
| WinSLIStartup | 29 |
| WinSLI | 30 |
| WinSLICleanup | 32 |
| Emisión de un verbo LUA | 33 |
| Información de SNA | 36 |
| Comprobación de BIND: RUI | 36 |
| Comprobación de BIND: SLI | 36 |
| Respuestas negativas y códigos de detección SNA | 37 |
| Ritmo | 38 |
| Segmentación. | 39 |
| Modificación de bits RH (cabecera de petición/respuesta) del sistema principal no estándar | 39 |
| Acuses de recibo de cortesía | 39 |
| Eliminación de datos hasta el final de la cadena | 39 |
| Información de SNA para RUI primaria | 40 |

| | |
|---|-----------|
| Responsabilidades de la aplicación RUI primaria | 40 |
| Ritmo | 41 |
| Segmentación. | 41 |
| Restricciones | 41 |
| Acuses de recibo de cortesía. | 41 |
| Eliminación de datos hasta el final de la cadena | 41 |
| Información de configuración | 42 |
| Control de enlace de datos (DLC), puerto y estación de enlace (LS). | 42 |
| LU | 42 |
| Agrupación de LU (opcional) | 42 |
| Consideraciones acerca de AIX o Linux | 43 |
| Archivo de cabecera LUA | 43 |
| Varios procesos y varias sesiones | 43 |
| Compilación y enlace de la aplicación LUA. | 44 |
| Consideraciones acerca de Windows | 44 |
| Múltiples sesiones y múltiples tareas | 44 |
| Compilación y enlace de programas LUA | 44 |
| Finalización de aplicaciones | 45 |
| Desarrollo de aplicaciones portables | 45 |
| Capítulo 3. Estructura de VCB LUA | 47 |
| Formato de VCB (bloque de control de verbos) LUA | 47 |
| Estructura de datos LUA_VERB_RECORD | 48 |
| Estructura de datos común | 48 |
| Estructura de datos específica | 54 |
| Capítulo 4. Verbos RUI | 59 |
| RUI_BID | 59 |
| Parámetros proporcionados | 59 |
| Parámetros devueltos | 60 |
| Interacción con otros verbos | 65 |
| Uso y restricciones | 65 |
| RUI_INIT | 66 |
| Parámetros proporcionados | 66 |
| Parámetros devueltos | 68 |
| Interacción con otros verbos | 72 |
| Uso y restricciones | 72 |
| RUI_INIT_PRIMARY | 73 |
| Parámetros proporcionados | 73 |
| Parámetros devueltos | 74 |
| Interacción con otros verbos | 76 |
| Uso y restricciones | 77 |
| RUI_PURGE | 77 |
| Parámetros proporcionados | 77 |
| Parámetros devueltos | 78 |
| Interacción con otros verbos | 81 |
| RUI_READ | 81 |
| Parámetros proporcionados | 82 |
| Parámetros devueltos | 83 |
| Interacción con otros verbos | 89 |
| Uso y restricciones | 89 |
| RUI_REINIT | 90 |
| Parámetros proporcionados | 90 |
| Parámetros devueltos | 91 |
| Interacción con otros verbos | 93 |
| Uso y restricciones | 93 |
| RUI_TERM | 94 |
| Parámetros proporcionados | 94 |
| Parámetros devueltos | 95 |
| Interacción con otros verbos | 98 |

| | |
|--|------------|
| RUI_WRITE | 98 |
| Parámetros proporcionados | 98 |
| Parámetros devueltos. | 100 |
| Interacción con otros verbos | 105 |
| Uso y restricciones | 105 |
| Capítulo 5. Verbos SLI | 107 |
| SLI_BID | 107 |
| Parámetros proporcionados. | 107 |
| Parámetros devueltos. | 108 |
| Interacción con otros verbos | 114 |
| Uso y restricciones. | 114 |
| SLI_CLOSE | 114 |
| Parámetros proporcionados. | 115 |
| Parámetros devueltos. | 116 |
| Interacción con otros verbos | 119 |
| Uso y restricciones | 120 |
| SLI_OPEN | 121 |
| Parámetros proporcionados. | 121 |
| Valor de retorno desde el punto de entrada SLI | 125 |
| Parámetros devueltos. | 125 |
| Interacción con otros verbos | 129 |
| Uso y restricciones | 130 |
| SLI_PURGE | 130 |
| Parámetros proporcionados. | 130 |
| Parámetros devueltos. | 131 |
| Interacción con otros verbos | 134 |
| SLI_RECEIVE | 134 |
| Parámetros proporcionados. | 134 |
| Parámetros devueltos. | 136 |
| Interacción con otros verbos | 143 |
| Uso y restricciones | 143 |
| SLI_SEND | 144 |
| Parámetros proporcionados. | 144 |
| Parámetros devueltos. | 146 |
| Interacción con otros verbos | 152 |
| Uso y restricciones | 152 |
| SLI_BIND_ROUTINE. | 153 |
| Parámetros proporcionados. | 153 |
| Parámetros devueltos. | 153 |
| Interacción con otros verbos | 154 |
| Uso y restricciones | 154 |
| SLI_SDT_ROUTINE | 154 |
| Parámetros proporcionados. | 154 |
| Parámetros devueltos. | 155 |
| Interacción con otros verbos | 155 |
| Uso y restricciones | 155 |
| SLI_STSN_ROUTINE. | 155 |
| Parámetros proporcionados. | 155 |
| Parámetros devueltos. | 156 |
| Interacción con otros verbos | 156 |
| Uso y restricciones | 156 |
| Capítulo 6. Aplicación LUA de ejemplo | 157 |
| Visión general del proceso | 157 |
| Cómo probar la aplicación | 158 |
| Requisitos del sistema principal | 159 |
| Configuración de la aplicación de ejemplo. | 159 |
| Compilación y enlace de la aplicación de ejemplo | 159 |
| Ejecución de la aplicación de ejemplo | 159 |

| | |
|---|------------|
| Apéndice A. Valores de código de retorno | 161 |
| Códigos de retorno primarios | 161 |
| Códigos de retorno secundarios | 161 |
| Apéndice B. Avisos | 165 |
| Marcas registradas. | 167 |
| Bibliografía | 169 |
| Publicaciones de Communications Server para Linux Versión 6.2.2 | 169 |
| Publicaciones de SNA (Systems Network Architecture) | 170 |
| Publicaciones de configuración de sistema principal | 171 |
| Publicaciones de z/OS Communications Server | 171 |
| Publicaciones sobre TCP/IP | 171 |
| Publicaciones sobre X.25. | 171 |
| Publicaciones de APPC | 171 |
| Publicaciones de programación | 171 |
| Otras publicaciones de gestión de red de IBM | 172 |
| Índice. | 173 |

Tablas

1. Convenios tipográficos. xii
2. Valores de parámetros de SLI_SEND según el tipo de mensaje 152

Figuras

| | | |
|----|---|-----|
| 1. | Componentes SNA utilizados para las comunicaciones LUA | 3 |
| 2. | Componentes SNA utilizados para las comunicaciones de RUI primaria. | 4 |
| 3. | Secuencia de comunicaciones RUI | 10 |
| 4. | Secuencia de comunicaciones SLI | 11 |
| 5. | Flujo de programa para la aplicación LUA de ejemplo | 158 |

Acerca de este manual

Esta publicación es una guía para desarrollar programas de aplicación de lenguaje C que utilizan la interfaz de LUA (aplicación de unidad lógica) convencional para comunicarse con un sistema principal SNA (arquitectura de red de sistemas). La implementación de IBM Communications Server para Linux de LUA se basa en la implementación de IBM de la RUI (Request/Response Unit Interface - Interfaz de unidad de petición/respuesta) en los productos OS/2 (por ejemplo **Communications Server para OS/2**), con modificaciones para el entorno de AIX o Linux.

IBM Communications Server para Linux es un producto de software de IBM que permite a un sistema que ejecuta Linux intercambie información con otros nodos de una red SNA.

Existen dos variantes de instalación diferentes de IBM Communications Server para Linux, en función del hardware en el que funciona:

Communications Server para Linux

Communications Server para Linux, programa producto número 5724-i33, funciona en lo siguiente:

- Estaciones de trabajo Intel de 32 bits que ejecutan Linux (i686)
- Estaciones de trabajo AMD64/Intel EM64T de 64 bits que ejecutan Linux (x86_64)
- Sistemas IBM pSeries que ejecutan Linux (ppc64)

Communications Server para Linux en System z

Communications Server para Linux en System z, programa producto número 5724-i34, funciona en sistemas principales System z que ejecutan Linux para System z (s390 o s390x).

En este manual, el nombre Communications Server para Linux se utiliza para indicar una de estas dos variantes y el término sistema "Communications Server para Linux" se utiliza para indicar cualquier tipo de sistema que ejecute Communications Server para Linux, excepto donde se describen explícitamente las diferencias.

Este manual se aplica a la Versión 6.2.2 de Communications Server para Linux.

A quién va dirigido este manual

Este manual está destinado a programadores expertos en C que escriben programas de transacciones SNA (Systems Network Architecture) para sistemas con Communications Server para Linux. Los programadores pueden tener experiencia previa con SNA o con los recursos de comunicaciones de Communications Server para Linux o pueden no tenerla.

Los programadores de aplicaciones diseñan y codifican programas de aplicación y de transacción que utilizan las interfaces de programación de Communications Server para Linux para enviar y recibir datos a través de una red SNA. Deben estar totalmente familiarizados con SNA, el programa remoto con el que se comunica el programación de aplicación o transacción y los entornos operativos y de programación de sistema operativo AIX o Linux.

A quién va dirigido este manual

Puede ver información más detallada acerca de cómo desarrollar programas de aplicación en la publicación de cada una de las API.

Cómo utilizar este manual

Este apartado describe cómo se organiza y presenta la información de este manual.

Organización de este manual

Este manual está organizado de la forma siguiente:

- El Capítulo 1, “Conceptos”, en la página 1 presenta los conceptos básicos de LUA. Está destinado a los programadores que no están familiarizados con LUA.
- El Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13 contiene la información de carácter general que necesita un programador al desarrollar aplicaciones LUA. Este capítulo también contiene información sobre los conceptos de SNA relevantes para el diseño de aplicaciones LUA y sobre cómo compilar y enlazar una aplicación LUA.
- El Capítulo 3, “Estructura de VCB LUA”, en la página 47, describe la estructura del VCB (Bloque de control de verbo) que se utiliza para todos los verbos LUA.
- El Capítulo 4, “Verbos RUI”, en la página 59, describe de forma detallada cada uno de los verbos RUI. Cada una de las descripciones ofrece la siguiente información: finalidad, formato de registro de verbo, parámetros suministrados, valores devueltos e información detallada sobre la interacción del verbo con otros verbos RUI.
- El Capítulo 5, “Verbos SLI”, en la página 107, describe de forma detallada cada uno de los verbos SLI. Cada una de las descripciones ofrece la siguiente información: finalidad, formato de registro de verbo, parámetros suministrados, valores devueltos e información detallada sobre la interacción del verbo con otros verbos SLI.
- El Capítulo 6, “Aplicación LUA de ejemplo”, en la página 157, describe la aplicación LUA de ejemplo de Communications Server para Linux que ilustra la utilización de los verbos RUI de LUA. Este capítulo también incluye instrucciones para compilar, enlazar y ejecutar la aplicación de ejemplo (incluidos los pasos de configuración de Communications Server para Linux necesarios).
- El Apéndice A, “Valores de código de retorno”, en la página 161, lista todos los códigos de retorno posibles de la interfaz de LUA en orden numérico y proporciona el significado de los mismos.

Convenios tipográficos

La Tabla 1 muestra los estilos tipográficos utilizados en esta publicación.

Tabla 1. Convenios tipográficos

| Elemento especial | Ejemplo de tipografía |
|---|--|
| Palabras enfatizadas | Realice una copia de seguridad de los archivos antes de suprimirlos |
| Título de publicación | <i>Communications Server para Linux, Guía de administración</i> |
| Nombre de vía de acceso o de archivo | <code>/usr/spool/uucp/miarch.bkp</code> |
| Programa o aplicación | <code>snaadmin</code> |
| Mandato o programa de utilidad de AIX / Linux | <code>define_node; cd</code> |

Tabla 1. Convenios tipográficos (continuación)

| Elemento especial | Ejemplo de tipografía |
|---|--|
| Referencia genérica a todos los mandatos de un tipo determinado | query_* (hace referencia a todos los mandatos de administración que consultan los detalles de un recurso) |
| Opción o indicador | -i |
| Parámetro o campo de Motif | <i>código de operación; Nombre de LU</i> |
| Valor literal o selección que el usuario puede entrar (incluidos los valores por omisión) | 255; Al iniciar el nodo |
| Constante o señal | AP_GET_LU_STATUS |
| Valor de retorno | AP_INVALID_FORMAT; 0; -1 |
| Variable que representa un valor suministrado | <i>nombre_archivo; nombre_LU; Id_usuario</i> |
| Variable de entorno | PATH |
| Verbo de programación | GET_LU_STATUS |
| Datos entrados por el usuario | 0p1 |
| Salida de la máquina | CERRAR |
| Función, llamada o punto de entrada | ioctl |
| Estructura de datos | termios |
| Tecla de 3270 | INTRO |
| Teclas del teclado | Control+D; Intro |
| Valor hexadecimal | 0x20 |

Convenios gráficos

AIX, LINUX

Este símbolo se utiliza para indicar el inicio de una sección de texto que se aplica sólo al sistema operativo AIX o Linux.

Se aplica a servidores Linux y IBM Remote API Client que se ejecuten en AIX, Linux, Linux para pSeries o Linux para System z.

WINDOWS

Este símbolo se utiliza para indicar el inicio de una sección de texto que se aplica a IBM Remote API Client en Windows.



Este símbolo indica el final de una sección de texto específica del sistema operativo. La información que viene a continuación de este símbolo se aplica independientemente del sistema operativo.

Publicaciones relacionadas

Para obtener información sobre SNA y LUA, consulte las siguientes publicaciones de IBM:

IBM Systems Network Architecture:

- *Technical Overview, GC30-3073*

Publicaciones relacionadas

- *Formats*, GA27-3136
- *Format and Protocol Reference Manual: Architectural Logic*, SC30-3112

Capítulo 1. Conceptos

En este capítulo presenta los conceptos básicos de LUA, la interfaz de programación de aplicaciones (API) de LU (unidad lógica) convencional.

Los temas tratados en este capítulo son los siguientes:

- ¿Qué es LUA?
- Elección de la interfaz a utilizar (RUI o SLI)
- LU y sesiones
- Verbos LUA
- Ejemplo de secuencia de comunicaciones LUA
- Compatibilidad con LUA

¿Qué es LUA?

LUA (la interfaz de programación de aplicaciones de LU convencional) es una API que le permite escribir aplicaciones de Communications Server para Linux a fin de comunicarse con aplicaciones de sistema principal.

La interfaz de LUA se proporciona a nivel de unidad de petición/respuesta (RU), permitiendo que el programador controle los mensajes SNA (Systems Network Architecture) enviados entre Communications Server para Linux y el sistema principal. Puede utilizarse para comunicarse con cualquiera de los tipos de LU: 0, 1, 2 ó 3 en el sistema principal; la aplicación es la encargada de enviar los mensajes SNA adecuados según lo requerido por la aplicación del sistema principal.

Por ejemplo, puede utilizar LUA para escribir un programa de emulación 3270 que se comunique con una aplicación 3270 de sistema principal; con Communications Server para Linux se incluye como aplicación LUA de ejemplo una versión simple de ésta y dicha versión se describe en el Capítulo 6, “Aplicación LUA de ejemplo”, en la página 157.

AIX, LINUX

Si el sistema Communications Server para Linux soporta la pasarela SNA para comunicaciones con las PU en sentido descendente, también puede escribir una aplicación LUA que actúe como la SNA primaria para las comunicaciones con las LU secundarias en estas PU en sentido descendente. Esto le permite emular una aplicación de sistema principal en el nodo de Communications Server para Linux o para descargar el proceso de una aplicación de sistema principal en el nodo de Communications Server para Linux. Esta función se describe como “RUI primaria”; es específica de Communications Server para Linux y es posible que otras implementaciones de LUA no la proporcionen.



Elección de la interfaz a utilizar

LUA incluye dos interfaces de programación en distintos niveles:

- La interfaz de unidad de petición (RUI) se proporciona a nivel de unidad de petición/respuesta (RU), permitiendo que el programador controle los mensajes SNA (Systems Network Architecture) enviados entre Communications Server para Linux y el sistema principal. La aplicación es la encargada de crear y enviar los mensajes SNA adecuados según lo requerido por la aplicación del sistema principal.

La interfaz RUI da soporte a los Perfiles de gestión de funciones SNA 2, 3, 4, 7 y 18, y a los Perfiles de servicios de transmisión SNA 2, 3, 4 y 7.

- La interfaz de nivel de sesión (SLI) es una interfaz de nivel superior que permite que el programador trabaje en un nivel de mensaje lógico en lugar de tratar con los detalles de las RU individuales. Por ejemplo:
 - La sesión se puede establecer y finalizar con un único verbo SLI (en lugar de con una secuencia de verbos RUI que correspondan a las RUI individuales implicadas en el inicio y la finalización de una sesión).
 - La biblioteca SLI controla el encadenamiento cuando la aplicación necesita enviar o recibir datos que superen la longitud de RU máxima especificada en BIND.
 - Para la mayoría de mandatos SNA que se envían al sistema principal, la biblioteca SLI puede crear la RU adecuada en la petición de la aplicación.

La interfaz SLI da soporte a los Perfiles de gestión de funciones SNA 3 y 4, y a los Perfiles de servicios de transmisión SNA 3 y 4.

Una aplicación tan solo puede utilizar una de estas interfaces para cada sesión. Por ejemplo, si empieza una sesión utilizando RUI, no puede emitir a continuación verbos SLI en dicha sesión.

Debe tener en cuenta los puntos siguientes antes de decidir qué API va a utilizar.

- SLI maneja algunos de los detalles de las RU individuales y su contenido, con lo que simplifica el proceso necesario en la aplicación. RUI requiere que la aplicación trate con cada RU de modo individual.
- RUI proporciona control sobre el contenido detallado de las RU que se envían al sistema principal y permite utilizar una amplia gama de perfiles de vinculación SNA. SLI no proporciona el mismo grado de control o flexibilidad.

AIX, LINUX

- La RUI incluye la RUI primaria (sólo AIX o Linux), que le permite escribir una aplicación que actúa como SNA primaria para las comunicaciones con las PU en sentido descendente. La interfaz SLI no proporciona esta función.



LU y sesiones

La Figura 1 en la página 3 muestra los componentes SNA utilizados para las comunicaciones LUA con un sistema principal .

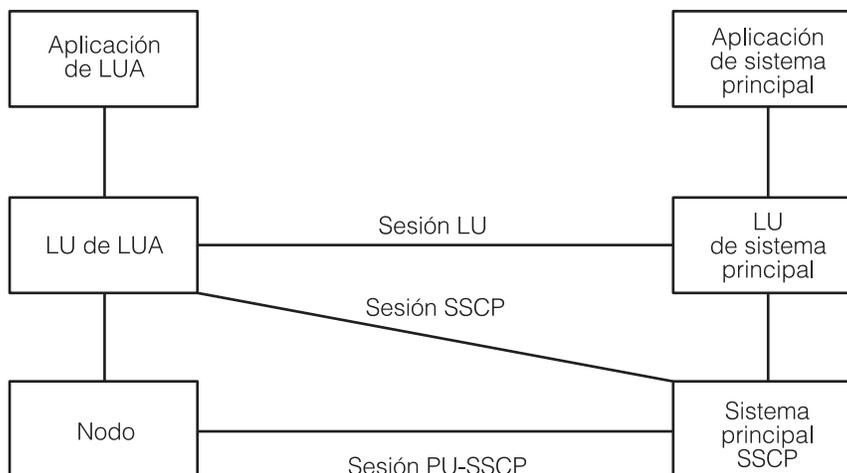


Figura 1. Componentes SNA utilizados para las comunicaciones LUA

Una aplicación LUA utiliza una LU de tipo 0-3 que se comunica con el sistema principal por medio del nodo de Communications Server para Linux. Existen tres sesiones entre el nodo de Communications Server para Linux y el nodo de sistema principal, como se indica a continuación:

- La sesión PU-SSCP (unidad física-punto de control de servicios del sistema), entre la PU 2.1 y el punto de control de servicios del sistema (SSCP) del sistema principal; se utiliza para controlar la unidad física (PU).
- La sesión de SSCP, entre la LU de Communications Server para Linux y el SSCP; se utiliza para controlar la LU.
- La sesión de LU, entre la LU de Communications Server para Linux y la LU de sistema principal; se utiliza para la transferencia de datos entre la LU y la aplicación de sistema principal.

La interfaz de programas de aplicación LUA permite que las aplicaciones envíen y reciban datos en la sesión de SSCP y en la sesión de LU. No proporciona acceso a la sesión PU-SSCP. Una aplicación LUA puede enviar datos en esta sesión utilizando el verbo TRANSFER_MS_DATA de MS (Management Services); para obtener más información, consulte la publicación *Communications Server for Linux, MS Programmer's Guide*.

WINDOWS

Para sistemas operativos Windows, TRANSFER_MS_DATA se proporciona como parte de la API de CSV (Common Service Verb); para obtener más información, consulte la publicación *Communications Server for Linux, CSV Programmer's Guide*.



AIX, LINUX

La Figura 2 muestra los componentes SNA utilizados para las comunicaciones LUA utilizando la RUI primaria en una LU en sentido descendente .

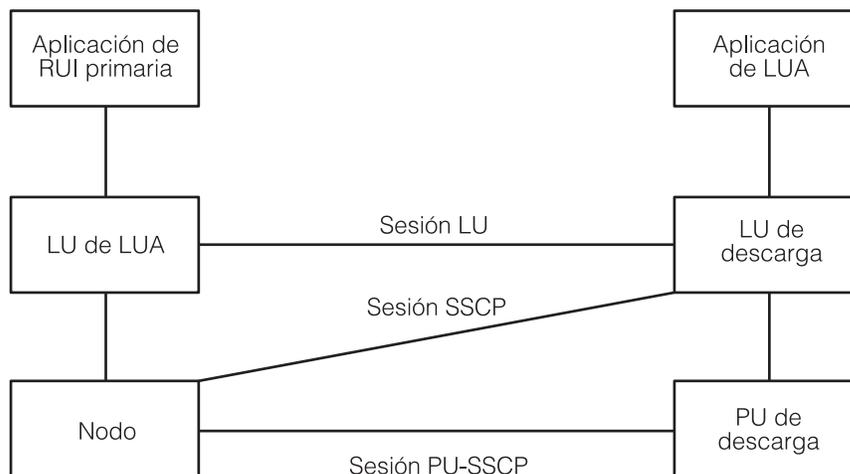


Figura 2. Componentes SNA utilizados para las comunicaciones de RUI primaria

Una aplicación RUI primaria utiliza una LU de tipo 0-3 que se comunica con la LU en sentido descendente por medio del nodo de Communications Server para Linux. Desde la perspectiva de la LU en sentido descendente, la LU de Communications Server para Linux actúa como LU de sistema principal y el nodo de Communications Server para Linux actúa como SSCP de sistema principal. Las tres sesiones entre estos componentes, y las restricciones en el acceso a estas sesiones, son equivalentes a las sesiones para una aplicación LUA que se comunique con un sistema principal.



Cada una de las sesiones de LU proporciona dos prioridades de mensajes: normal y urgente. Los mensajes urgentes o de flujo rápido tienen preferencia respecto a otros mensajes que están en espera de ser transmitidos en la misma sesión. Existen cuatro flujos en los que puede enviarse o recibirse un mensaje:

- Sesión de SSCP de flujo rápido
- Sesión de LU de flujo rápido
- Sesión de SSCP de flujo normal
- Sesión de LU de flujo normal

El flujo normal de una sesión de LU transporta datos de aplicación; los otros flujos se utilizan para los mensajes de control y el arranque.

La implementación de Communications Server para Linux de LUA no habilita las aplicaciones para que envíen datos en el flujo acelerado de SSCP y no devuelven datos a una aplicación en este flujo.

Configuración

Cada LU utilizada por una aplicación LUA se debe configurar utilizando el programa de administración de Motif, el programa de administración de línea de

mandatos o la API de NOF (recurso de operador de nodo) (para obtener más información, consulte la publicación *Communications Server para Linux, Guía de administración* o la publicación *Communications Server for Linux, NOF Programmer's Guide*). Además, la configuración de Communications Server para Linux puede incluir agrupaciones de LU. Una agrupación es un grupo de LU que tienen características similares, de modo que una aplicación puede utilizar cualquier LU libre del grupo. Esto puede utilizarse para asignar las LU por orden de llegada cuando hay más aplicaciones que LU disponibles o para proporcionar varias LU en diferentes conexiones.

Verbos LUA

Una aplicación accede a LUA mediante los verbos LUA. Cada verbo proporciona parámetros a LUA, que realiza la función solicitada y devuelve parámetros a la aplicación.

Resumen de verbos RUI

La lista siguiente contiene un breve resumen de cada uno de los verbos RUI LUA (para obtener una explicación detallada de cada verbo, consulte el Capítulo 4, "Verbos RUI", en la página 59):

RUI_BID

Este verbo permite que la aplicación determine cuándo está disponible para leerse la información del sistema principal.

RUI_INIT

Este verbo prepara la sesión de SSCP para una aplicación LUA.

AIX, LINUX

RUI_INIT_PRIMARY

Este verbo configura la sesión SSCP para una aplicación LUA que actúa como SNA primaria para comunicaciones con una LU en sentido descendente.

■

RUI_PURGE

Este verbo cancela un verbo RUI_READ pendiente.

RUI_READ

Este verbo recibe datos o información de estado enviada desde el sistema principal a la LU de la aplicación LUA, en la sesión de SSCP o en la sesión de LU.

AIX, LINUX

RUI_REINIT

Este verbo vuelve a establecer la sesión de SSCP para una aplicación LUA tras una sesión fallida. Lo utilizan las aplicaciones que estaban utilizando una LU desde una agrupación y que necesitan restablecer la sesión utilizando la misma LU a fin de continuar su proceso.

■

RUI_TERM

Este verbo finaliza la sesión de SSCP para una aplicación LUA. También finaliza la sesión de LU si estaba activa.

RUI_WRITE

Este verbo envía datos al sistema principal en la sesión de SSCP o en la sesión de LU.

Resumen de verbos SLI

La lista siguiente contiene un breve resumen de cada uno de los verbos SLI LUA (para obtener una explicación detallada de cada verbo, consulte el Capítulo 5, "Verbos SLI", en la página 107):

SLI_BID

Este verbo permite que la aplicación determine cuándo está disponible para leerse la información del sistema principal.

SLI_CLOSE

Este verbo finaliza la sesión para una aplicación LUA.

SLI_OPEN

Este verbo configura la sesión para una aplicación LUA.

SLI_PURGE

Este verbo cancela un verbo SLI_RECEIVE pendiente.

SLI_RECEIVE

Este verbo recibe datos o información de estado enviada desde el sistema principal a la LU de la aplicación LUA, en la sesión de SSCP o en la sesión de LU.

SLI_SEND

Este verbo envía datos al sistema principal en la sesión de SSCP o en la sesión de LU.

En el verbo SLI_OPEN, la aplicación puede especificar opcionalmente las direcciones de sus propias rutinas para procesar peticiones de BIND, STSN y SDT desde el sistema principal. Si proporciona estas rutinas y llega una petición del tipo adecuado desde el sistema principal, LUA envía un verbo adicional a la rutina proporcionada por la aplicación adecuada para permitir que procese la petición, tal como se describe a continuación.

SLI_BIND_ROUTINE

LUA envía este verbo a la rutina BIND proporcionada por la aplicación cuando llega una petición de BIND desde el sistema principal. La aplicación puede aceptar BIND, negociar parámetros de BIND o rechazar BIND tal como se describe en el apartado "Información de SNA" en la página 36.

Si la aplicación no proporciona ninguna rutina BIND, LUA realiza una comprobación limitada de BIND y responde al sistema principal del modo adecuado.

SLI_STSN_ROUTINE

LUA envía este verbo a la rutina STSN proporcionada por la aplicación cuando llega una petición de STSN desde el sistema principal. La aplicación puede responder a STSN o rechazarla con un código de detección SNA adecuado, tal como se describe en el apartado "Información de SNA" en la página 36.

Si la aplicación no proporciona ninguna rutina STSN, LUA devuelve una respuesta afirmativa que indica que no hay datos disponibles.

SLI_SDT_ROUTINE

LUA envía este verbo a la rutina SDT proporcionada por la aplicación cuando llega una petición de SDT desde el sistema principal. La aplicación puede responder a SDT o rechazarla con un código de detección SNA adecuado, tal como se describe en el apartado “Información de SNA” en la página 36.

Si la aplicación no proporciona ninguna rutina SDT, LUA devuelve una respuesta afirmativa.

Finalización asíncrona de verbos

Algunos verbos LUA finalizan rápidamente, tras algún proceso local (por ejemplo, el verbo RUI_PURGE); no obstante, la mayoría de los verbos tardan algún tiempo en finalizar, porque requieren que se envíen mensajes al nodo o a la aplicación del sistema principal, y que éstos los reciban. Por ello, LUA se implementa como una interfaz asíncrona; puede devolverse el control a la aplicación mientras hay un verbo en proceso, de modo que la aplicación pueda seguir procesando (incluso emitir otros verbos LUA).

AIX, LINUX

Una vez finalizado el verbo, LUA llama a una rutina de devolución de llamada proporcionada por la aplicación. Esta rutina puede realizar más procesos en los datos devueltos, emitir más verbos LUA o simplemente actuar como un indicador de que el verbo ha finalizado.

- Los verbos RUI se pueden finalizar de manera síncrona u asíncrona. La aplicación debe comprobar el código de retorno primario del VCB para determinar qué modalidad de finalización se aplica para cada verbo.
- Los verbos SLI finalizan siempre en modalidad asíncrona. Después de emitir el verbo, la aplicación no debe acceder al VCB hasta que se haya llamado a su rutina de devolución de llamada. Puede procesar el VCB desde dentro de la rutina de devolución de llamada o desde el subproceso de ejecución principal del programa tras haber finalizado la rutina de devolución de llamada.

WINDOWS

Cuando el verbo finaliza, LUA emite un mensaje a un manejador de ventanas proporcionado por la aplicación o señala un manejador de sucesos proporcionado por la aplicación.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Ejemplo de secuencia de comunicaciones LUA

La Figura 3 en la página 10, muestra un ejemplo de secuencia de comunicaciones LUA en la que se utilizan verbos RUI y la Figura 4 en la página 11 muestra la secuencia equivalente en la que se utilizan verbos SLI.

En el ejemplo de RUI, la aplicación realiza los pasos siguientes:

1. Emite el verbo RUI_INIT para establecer la sesión de SSCP. El verbo RUI_INIT no se completa hasta que Communications Server para Linux ha recibido un mensaje de activación de unidad lógica (ACTLU) del sistema principal y ha enviado una respuesta positiva; sin embargo, Communications Server para Linux maneja estos mensajes y éstos no se exponen a la aplicación LUA.
2. Envía un mensaje INITSELF al SSCP para solicitar un BIND y lee la respuesta.
3. Lee un mensaje BIND del sistema principal y graba la respuesta. Esto establece la sesión de LU.
4. Lee un mensaje SDT del sistema principal, que indica que la inicialización ha finalizado y que puede iniciarse la transferencia de datos.
5. Envía una cadena de datos que consta de tres RU (la última indica que se requiere una respuesta concreta) y lee la respuesta.
6. Lee una cadena de datos que consta de dos RU y graba la respuesta.
7. Lee un mensaje UNBIND del sistema principal y graba la respuesta. Esto finaliza la sesión de LU.
8. Emite el verbo RUI_TERM para finalizar la sesión de SSCP. (Communications Server para Linux envía un mensaje NOTIFY al sistema principal y espera una respuesta positiva; sin embargo, Communications Server para Linux maneja estos mensajes y éstos no se exponen a la aplicación LUA.)

El ejemplo de SLI muestra la misma secuencia de mensajes circulando entre el sistema principal y la aplicación. Los verbos SLI usados son similares a los que se usan en el ejemplo de RUI, pero deben considerarse las siguientes diferencias:

- SLI_OPEN maneja completamente la inicialización de sesión; la aplicación no necesita leer y grabar cada RU individual en la secuencia de inicialización como en el ejemplo de RUI.
- LUA utiliza las rutinas BIND y SDT de la aplicación (especificadas en SLI_OPEN) para permitir que la aplicación procese los mensajes BIND y SDT del sistema principal. Estas rutinas deben tener un retorno síncrono. Los demás verbos SLI finalizan de modo asíncrono.
- SLI_RECEIVE y SLI_SEND manejan cadenas completas de datos, de modo que la aplicación tan solo necesita un verbo para recibir o enviar los datos aunque tengan una longitud suficiente para necesitar dos o tres RU. (En el ejemplo de RUI, la aplicación debe recibir o enviar cada RU con un verbo separado.)

Ejemplo de secuencia de comunicaciones LUA

La lista siguiente muestra las abreviaturas que se utilizan en la Figura 3 en la página 10 y la Figura 4 en la página 11.

| | |
|-------------|---|
| SSCP normal | Sesión de SSCP de flujo normal |
| LU normal | Sesión de LU de flujo normal |
| LU rápida | Sesión de LU de flujo rápido |
| +respuesta | Respuesta afirmativa al mensaje indicado |
| BC | Iniciar cadena |
| MC | Mitad de la cadena |
| EC | Finalizar cadena |
| CD | Indicador de cambio de dirección activado |
| RQD | Respuesta concreta necesaria |

La Figura 3 en la página 10 muestra los verbos RUI utilizados para iniciar una sesión, intercambiar datos y finalizar la sesión, así como los mensajes SNA enviados y recibidos. Las flechas indican la dirección del flujo de los mensajes SNA.

Ejemplo de secuencia de comunicaciones LUA

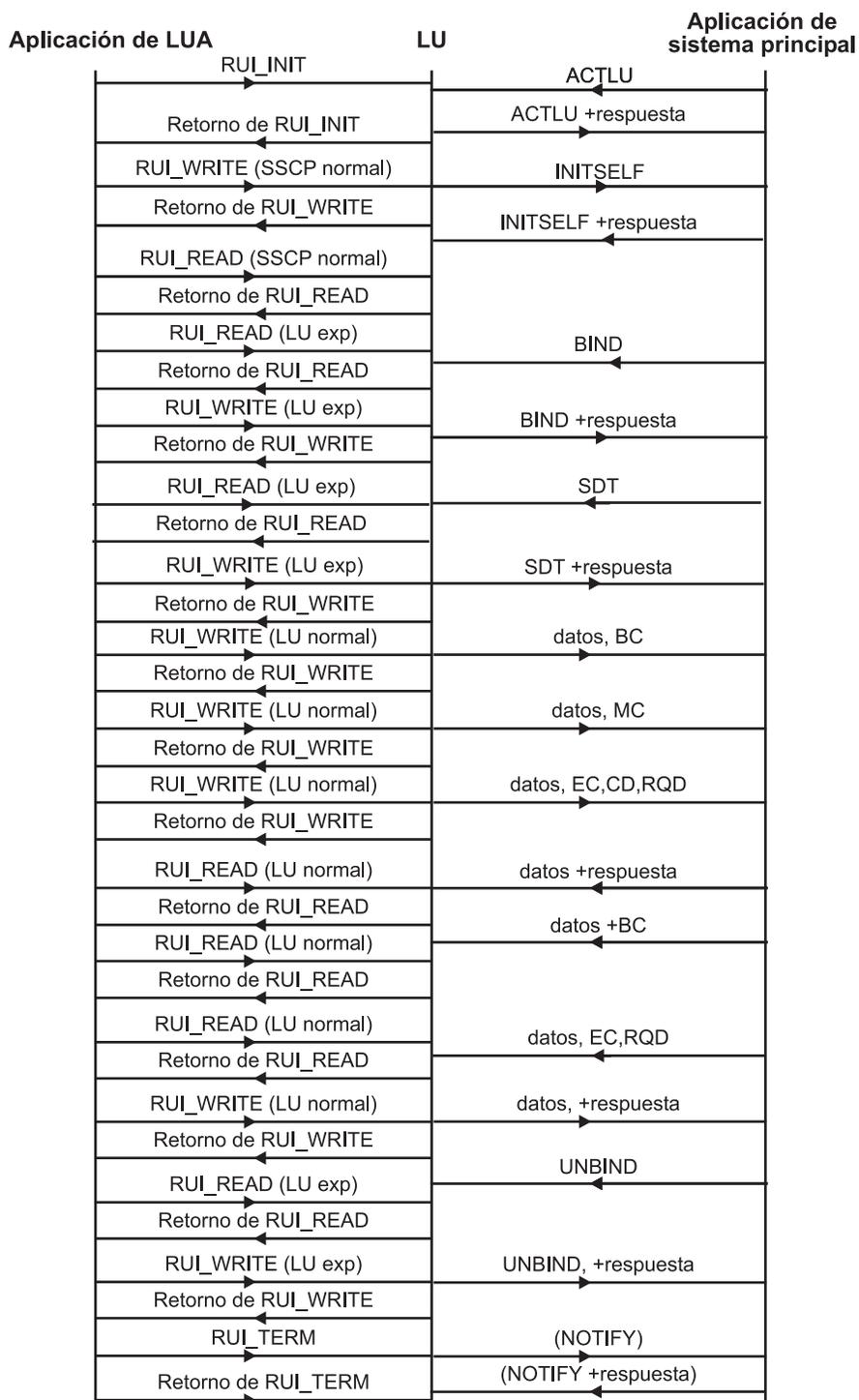


Figura 3. Secuencia de comunicaciones RUI

Ejemplo de secuencia de comunicaciones LUA

La Figura 4 muestra los verbos SLI equivalentes que se utilizan para la misma secuencia de mensajes SNA.

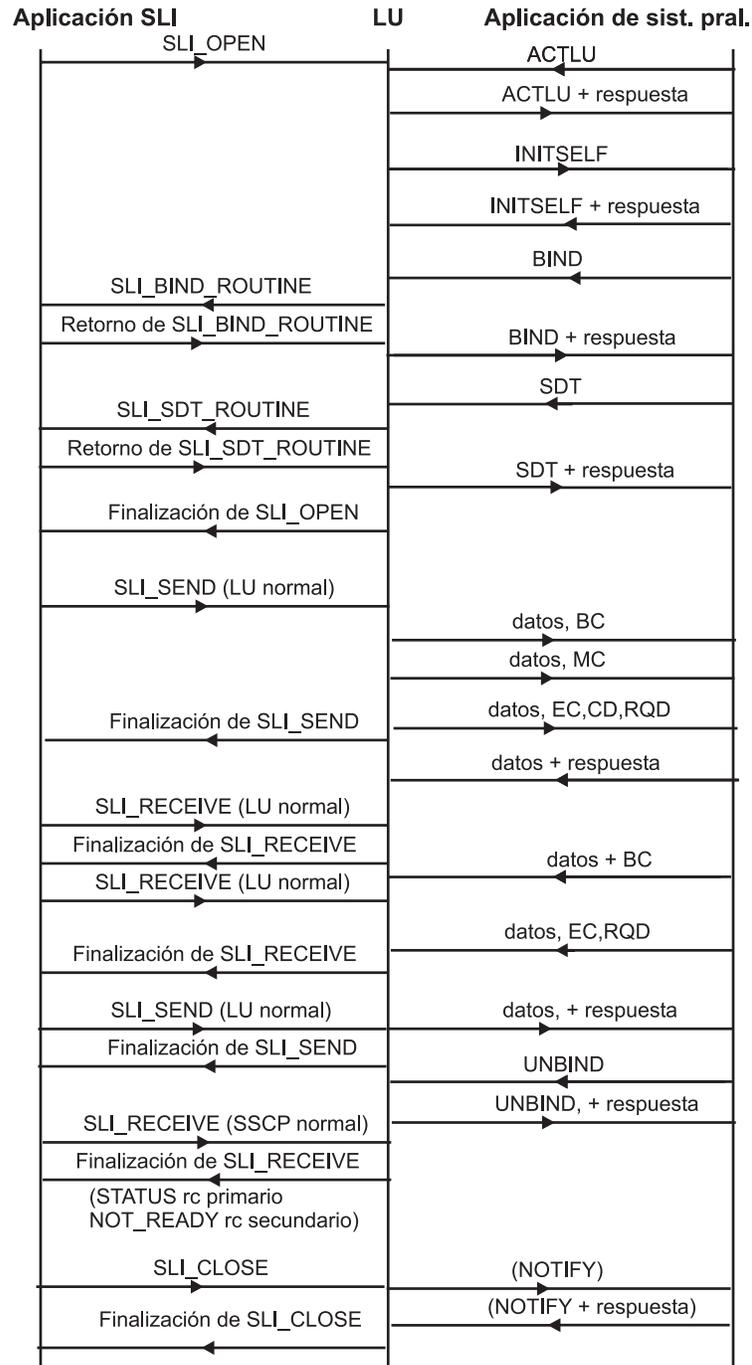


Figura 4. Secuencia de comunicaciones SLI

Compatibilidad con LUA

AIX, LINUX

Los verbos RUI_INIT_PRIMARY y RUI_REINIT son ampliaciones de la especificación de interfaz de LU estándar. No están disponibles en un Remote API Client en Windows y es posible que no estén disponibles en otras implementaciones de LUA.

WINDOWS

La implementación de LUA en Remote API Client en Windows se ha diseñado para que sea compatible con LUA de Windows (como define la especificación de WOSA SNA); se pueden utilizar aplicaciones escritas para LUA de Windows con Remote API Client sin ninguna modificación.



Capítulo 2. Diseño y desarrollo de aplicaciones LUA

La información de este capítulo le ayudará a desarrollar programas de aplicación LUA. Se tratan los temas siguientes:

- Puntos de entrada LUA para aplicaciones AIX o Linux
- Puntos de entrada LUA para aplicaciones Windows
- Emisión de un verbo LUA
- Información de SNA
- Información de configuración
- Consideraciones acerca de AIX o Linux
- Consideraciones acerca de Windows
- Desarrollo de aplicaciones portables

Puntos de entrada LUA para aplicaciones AIX o Linux

AIX, LINUX

Las aplicaciones que se ejecutan en AIX o Linux acceden a LUA utilizando la llamada de función RUI o SLI, especificando la dirección de un VCB (Bloque de control de verbos) que contiene información para un verbo LUA. Communications Server para Linux devuelve el control a la aplicación inmediatamente.

El VCB devuelto contiene un valor que indica si el proceso del verbo todavía se está ejecutando o ha finalizado.

- En algunos casos, el verbo aún se está procesando cuando se devuelve el control a la aplicación; entonces Communications Server para Linux utiliza una rutina de devolución de llamada proporcionada por la aplicación para devolver los resultados del proceso de verbo.
- En otros casos, el proceso de verbo se ha completado cuando Communications Server para Linux devuelve el control a la aplicación; Communications Server para Linux no utiliza la rutina de devolución de llamada de la aplicación. Esto se aplica en particular si el verbo no pasa satisfactoriamente las comprobaciones de parámetros iniciales de LUA o las comprobaciones de estado y, en consecuencia, no se puede actuar en el mismo.
- Para SLI_OPEN, si las comprobaciones iniciales son satisfactorias, la llamada de función SLI devuelve un valor distinto de cero que representa el ID de sesión de la sesión nueva. A continuación, Communications Server para Linux utiliza la rutina de devolución de llamada proporcionada por la aplicación del mismo modo que para otros verbos. La aplicación puede utilizar el nuevo ID de sesión para emitir un rango limitado de verbos subsiguientes en la sesión, sin esperar a que se llame a la rutina de devolución de llamada. Para obtener detalles sobre qué verbos se pueden emitir en esta situación, consulte “Interacción con otros verbos” en la página 129.

Nota: Debido al modo en que funcionan las rutinas de devolución de llamada del sistema operativo, es posible que se llame a la rutina de devolución de llamada de la aplicación antes de que se devuelva el control a la aplicación desde la llamada de función inicial para el verbo. Esto significa que, si la rutina de devolución de llamada modifica o suprime el VCB devuelto, es

Puntos de entrada LUA para aplicaciones AIX o Linux

posible que el subproceso de ejecución principal del programa no pueda comprobar los parámetros de VCB para determinar que el verbo funciona de modo asíncrono. Puede que deba tener en cuenta esto en el diseño de la aplicación.

Los puntos de entrada RUI y SLI se definen en el archivo de cabecera de LUA `/usr/include/sna/lua_c.h` (AIX) o bien `/opt/ibm/sna/include/lua_c.h` (Linux).

Llamada de función RUI

```
void RUI(verb)
LUA_VERB_RECORD * verb;
```

Llamada de función SLI

```
AP_UINT32 SLI(verb)
LUA_VERB_RECORD * verb;
```

Parámetros proporcionados

El parámetro proporcionado es:

verb Puntero a un bloque de control de verbos (VCB), que contiene los parámetros para que se emita el verbo. La estructura de VCB se define en el archivo de cabecera de LUA `lua_c.h` y se describe en el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice `memset`:

```
memset(vcb, 0, sizeof(vcb));
```

Valores devueltos

Para RUI, y para todos los verbos SLI salvo para `SLI_OPEN`, el punto de entrada no devuelve ningún valor. Los parámetros devueltos en el VCB indican si el verbo ha finalizado de modo síncrono o si finalizará de modo asíncrono; una vez finalizado el verbo, el VCB contiene los resultados del verbo.

Para `SLI_OPEN`, el punto de entrada devuelve un valor que indica si el VCB ha pasado las comprobaciones iniciales de LUA:

- Un valor de retorno de 0 (cero) indica que el verbo no ha pasado correctamente las comprobaciones iniciales de LUA (por ejemplo, porque la aplicación ha proporcionado parámetros incorrectos). Communications Server para Linux no llamará a la rutina de devolución de llamada proporcionada por la aplicación.

- Un valor distinto de cero representa el ID de sesión de la sesión nueva que SLI_OPEN iniciará.
Este valor de retorno no indica que el verbo ha finalizado. Communications Server para Linux llamará a la rutina de devolución de llamada proporcionada por la aplicación para indicar la finalización de SLI_OPEN cuando se haya configurado la sesión.

Para ver más información, consulte “Uso”.

Uso

A veces LUA puede finalizar todo el proceso para un verbo en cuanto se emite. Esto se aplica en particular si el verbo no pasa satisfactoriamente las comprobaciones de parámetros iniciales de LUA o las comprobaciones de estado y, en consecuencia, no se puede actuar en el mismo. En este caso, el verbo vuelve de manera síncrona; el código de retorno primario está definido en un valor distinto de LUA_IN_PROGRESS, y el bit *lua_flag2.async* tiene el valor 0 (cero). (Para obtener información sobre estos parámetros devueltos, consulte el Capítulo 4, “Verbos RUI”, en la página 59 o el Capítulo 5, “Verbos SLI”, en la página 107.)

Otras veces, LUA debe esperar la información del nodo o la LU remota para poder completar el verbo. En tal caso, el verbo se devuelve de manera asíncrona; el código de retorno primario se establece en LUA_IN_PROGRESS y el bit *lua_flag2.async* es 1. Ahora la aplicación puede realizar otro proceso o bien esperar la notificación de LUA de que el verbo ha finalizado. LUA emite esta notificación estableciendo el código de retorno primario en su valor final, dejando el valor de *lua_flag2.async* en 1.

Como parte del VCB proporcionado, la aplicación suministra un puntero a una rutina de devolución de llamada (en el parámetro *lua_post_handle*). Si el verbo finaliza de manera síncrona, LUA no llama a la rutina de devolución de llamada. Si finaliza de manera asíncrona, LUA indica la finalización del verbo llamando a la rutina de devolución de llamada con un parámetro (un puntero al bloque de control de verbos original). Para ver más información, consulte “Rutina de devolución de llamada para la finalización asíncrona de verbos” en la página 16.

Nota:

1. Una aplicación no puede predecir si un determinado verbo finalizará de manera síncrona o asíncrona.
2. Si el parámetro *lua_flag2.async* indica que el verbo finalizará de manera asíncrona, el subproceso de ejecución principal del programa no debería acceder a ningún otro parámetro del VCB en este punto. Cuando LUA llama a la rutina de devolución de llamada, la aplicación sí puede acceder a los parámetros del VCB.
3. Debido al modo en que funcionan las rutinas de devolución de llamada del sistema operativo, es posible que se llame a la rutina de devolución de llamada de la aplicación antes de que se devuelva el control a la aplicación desde la llamada de función inicial para el verbo. Esto significa que, si la rutina de devolución de llamada modifica o suprime el VCB devuelto, es posible que el subproceso de ejecución principal del programa no pueda comprobar los parámetros de VCB para determinar que el verbo funciona de modo asíncrono. Puede que deba tener en cuenta esto en el diseño de la aplicación.

Rutina de devolución de llamada para la finalización asíncrona de verbos

Para que un verbo LUA finalice de manera asíncrona, la aplicación debe suministrar un puntero a una rutina de devolución de llamada. Este apartado describe cómo utiliza Communications Server para Linux esta rutina y las funciones que debe realizar.

Llamada de función

```
void callback(verb)
LUA_VERB_RECORD * verb;
{
    .
    .
}
```

Parámetros proporcionados

Communications Server para Linux llama a la rutina con el siguiente parámetro:

verb Puntero al VCB proporcionado por la aplicación, incluyendo los parámetros devueltos establecidos por Communications Server para Linux. La rutina de devolución de llamada puede ejecutar todos los procesos necesarios en los parámetros devueltos del VCB o simplemente establecer una variable para informar al programa principal de que el verbo ha finalizado.

Nota: Debido al modo en que funcionan las rutinas de devolución de llamada del sistema operativo, es posible que se llame a la rutina de devolución de llamada de la aplicación antes de que se devuelva el control a la aplicación desde la llamada de función inicial para el verbo. Esto significa que, si la rutina de devolución de llamada modifica o suprime el VCB devuelto, es posible que el subproceso de ejecución principal del programa no pueda comprobar los parámetros de VCB para determinar que el verbo funciona de modo asíncrono. Puede que deba tener en cuenta esto en el diseño de la aplicación.

Valores devueltos

No se devuelve ningún valor.

Puntos de entrada LUA para aplicaciones Windows

WINDOWS

Una aplicación Windows accede a LUA utilizando las funciones siguientes:

RUI Emite un verbo RUI. Si el verbo finaliza de manera asíncrona, LUA indica la finalización señalando un manejador de sucesos proporcionado por la aplicación.

WinRUIStartup

Registra la aplicación como un usuario de RUI de Windows y determina si el software de LUA da soporte al nivel de función necesario para la aplicación.

WinRUI Emite un verbo RUI. Si el verbo finaliza de modo asíncrono, LUA indicará la finalización emitiendo un mensaje en la ventana de la aplicación.

WinRUIGetLastInitStatus

Comprueba el estado de una sesión RUI (iniciada por un verbo RUI_INIT

Puntos de entrada LUA para aplicaciones Windows

anterior que todavía está pendiente), solicita notificación de los cambios realizados en el estado de la sesión o cancela esta notificación.

WinRUICleanup

Elimina el registro de la aplicación cuando termina de utilizar RUI .

GetLuaReturnCode

Genera una cadena de caracteres imprimibles para los códigos de retorno primario y secundario obtenidos en un verbo LUA.

SLI Emite un verbo SLI. Si el verbo finaliza de manera asíncrona, LUA indica la finalización señalando un manejador de sucesos proporcionado por la aplicación.

WinSLIStartup

Registra la aplicación como un usuario de SLI de Windows y determina si el software de LUA da soporte al nivel de función necesario para la aplicación.

WinSLI Emite un verbo SLI. Si el verbo finaliza de modo asíncrono, LUA indicará la finalización emitiendo un mensaje en la ventana de la aplicación.

WinSLICleanup

Elimina el registro de la aplicación cuando termina de utilizar SLI .

Una aplicación RUI debe llamar a `WinRUIStartup` antes de intentar emitir cualquier verbo LUA utilizando la llamada a `WinRUI`.

Mientras un verbo `RUI_INIT` está pendiente, la aplicación puede utilizar `WinRUIGetLastInitStatus` para determinar el estado de la sesión LUA iniciada por este verbo; a continuación, puede cancelar el verbo `RUI_INIT` si es necesario. La función `WinRUIGetLastInitStatus` se puede utilizar para comprobar el estado actual sin solicitar la notificación de los cambios subsiguientes, para solicitar notificación asíncrona de los cambios subsiguientes realizados en el estado de la sesión o para cancelar una petición anterior de notificación de cambios en el estado.

Si un verbo devuelve códigos de retorno non-LUA_OK, la aplicación puede utilizar `GetLuaReturnCode` para obtener una representación de una cadena de texto de estos códigos de retorno, que se puede utilizar para generar mensajes de error estándar.

Cuando termina de emitir verbos LUA utilizando la llamada a `WinRUI`, debe llamar a `WinRUICleanup` antes de finalizar; no debe intentar emitir ningún otro verbo RUI después de llamar a `WinRUICleanup`.

Una aplicación SLI debe llamar a `WinSLIStartup` antes de intentar emitir cualquier verbo LUA utilizando la llamada a `WinSLI`.

Si un verbo devuelve códigos de retorno non-LUA_OK, la aplicación puede utilizar `GetLuaReturnCode` para obtener una representación de una cadena de texto de estos códigos de retorno, que se puede utilizar para generar mensajes de error estándar.

Cuando termina de emitir verbos LUA utilizando la llamada a `WinSLI`, debe llamar a `WinSLICleanup` antes de finalizar; no debe intentar emitir ningún otro verbo SLI después de llamar a `WinSLICleanup`.

Los apartados siguientes describen estas funciones.

RUI

La aplicación utiliza esta función para emitir un verbo RUI LUA. Si el verbo finaliza de manera asíncrona, LUA indica la finalización señalando un manejador de sucesos proporcionado por la aplicación.

No es necesario que la aplicación emita un verbo WinRUIStartup antes de realizar esta llamada.

Llamada de función

```
void WINAPI RUI(verb)
LUA_VERB_RECORD FAR * verb;
```

Parámetros proporcionados

El parámetro proporcionado es:

verb Puntero a un bloque de control de verbos (VCB), que contiene los parámetros para que se emita el verbo. La estructura de VCB está definida en el archivo de cabecera de LUA **winlua.h**; este archivo está instalado en el subdirectorio **\sdk** para aplicaciones de 32 bits o **\sdk64** para aplicaciones de 64 bits, en el directorio donde ha instalado el software Windows Client. Para obtener una explicación sobre la estructura de VCB, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice `memset`:

```
memset(vcb, 0, sizeof(vcb));
```

Valores devueltos

El punto de entrada no devuelve ningún valor. Cuando la llamada vuelve, la aplicación puede examinar los parámetros del VCB para determinar si el verbo ha finalizado de manera síncrona o bien finalizará de manera asíncrona. Para ver más información, consulte “Uso”.

Uso

A veces LUA puede realizar todo el proceso de un verbo en cuanto se emite. En este caso, el verbo vuelve de manera síncrona; el código de retorno primario está definido en un valor distinto de `LUA_IN_PROGRESS`, y el bit `lua_flag2.async` tiene el valor 0 (cero). (Para ver información sobre estos parámetros devueltos, consulte el Capítulo 4, “Verbos RUI”, en la página 59.)

Otras veces, LUA debe esperar la información del nodo o la LU remota para poder completar el verbo. En tal caso, el verbo se devuelve de manera asíncrona; el

Puntos de entrada LUA para aplicaciones Windows

código de retorno primario se establece en `LUA_IN_PROGRESS` y el bit `lua_flag2.async` es 1. Ahora la aplicación puede realizar otro proceso o bien esperar la notificación de LUA de que el verbo ha finalizado. LUA emite esta notificación estableciendo el código de retorno primario en su valor final, dejando el bit de `lua_flag2.async` en 1.

Como parte del VCB proporcionado, la aplicación suministra un manejador de sucesos en el parámetro `lua_post_handle`. El suceso debe estar en estado no señalado y el manejador debe tener acceso `EVENT_MODIFY_STATE` al suceso. Si el verbo finaliza de manera síncrona, LUA no señala este manejador de sucesos. Si el verbo finaliza de manera asíncrona, LUA indica la finalización del verbo señalando el manejador de sucesos.

La aplicación emite una llamada a `WaitForSingleObject` o `WaitForMultipleObject` para esperar el manejador de sucesos. Cuando se señala el suceso, la aplicación examina el código de retorno primario y el código de retorno secundario para buscar errores.

Una aplicación no puede predecir si un determinado verbo finalizará de manera síncrona o asíncrona.

WinRUIStartup

La aplicación utiliza esta función para registrarse como un usuario de RUI de Windows y para determinar si el software de LUA da soporte a la versión de LUA de Windows que necesita.

Llamada de función

```
int WINAPI WinRUIStartup (
    WORD wVersionRequired;
    LUADATA far * lpData;
)

typedef struct
{
    WORD wVersion;
    char szDescription[41];
} LUADATA;
```

Parámetros proporcionados

El parámetro proporcionado es:

wVersionRequired

La versión de LUA de Windows que la aplicación necesita.
Communications Server para Linux soporta la Versión 1.0.

El byte de orden inferior especifica el número de versión más alto y el byte de orden superior especifica el número de versión más bajo. Por ejemplo:

| Versión | wVersionRequired |
|---------|------------------|
| 1.0 | 0x0001 |
| 1.1 | 0x0101 |
| 2.0 | 0x0002 |

Si la aplicación puede utilizar más de una versión, debe especificar la versión más alta que puede utilizar.

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

Puntos de entrada LUA para aplicaciones Windows

0 (cero)

La aplicación se ha registrado satisfactoriamente y el software de LUA de Windows da soporte al número de versión especificado por la aplicación o a una versión inferior. La aplicación debería comprobar el número de versión en la estructura LUADATA para asegurarse de que es lo suficientemente alto.

WLUAVERNOTSUPPORTED

El número de versión especificado por la aplicación no está soportado por el software de LUA de Windows. La aplicación no se ha registrado.

WLUAINITREJECT

La aplicación ya ha llamado a WinRUIStartup y se ha registrado satisfactoriamente. No debe llamarse a esta función más de una vez.

WLUASYSNOTREADY

El software Communications Server para Linux no se ha iniciado o el nodo local no está activo. La aplicación no se ha registrado.

WLUAFAILURE

Se ha producido un error del sistema operativo durante la inicialización del software de LUA de Windows. La aplicación no se ha registrado. Compruebe los archivos de anotaciones cronológicas para ver si existen mensajes que indiquen la causa de la anomalía.

Si el valor de retorno de WinRUIStartup es 0 (cero), la estructura LUADATA contiene información sobre el soporte proporcionado por el software de LUA de Windows. Si el valor de retorno es distinto de cero, el contenido de esta estructura no está definido y la aplicación no debería comprobarlo. Los parámetros de esta estructura son los siguientes:

wVersion

El número de versión de LUA de Windows al cual da soporte el software, en el mismo formato que el parámetro *wVersionRequired* (consulte "Parámetros proporcionados" en la página 19). Communications Server para Linux soporta la Versión 1.0.

Si el software da soporte al número de versión solicitado, este parámetro se establece en el mismo valor que el parámetro *wVersionRequired*; de lo contrario, se establece en la versión más alta a la que dé soporte el software, que será inferior al número de versión proporcionado por la aplicación. La aplicación debe comprobar el valor devuelto y realizar la acción que se indica a continuación:

- Si el número de versión devuelto es igual que el número de versión solicitado, la aplicación puede utilizar esta implementación de LUA de Windows.
- Si el número de versión devuelto es inferior al número de versión solicitado, la aplicación puede utilizar esta implementación de LUA de Windows pero no debe intentar utilizar características no soportadas por el número de versión devuelto. Si esto no es posible dado que necesita características que están disponibles en la versión inferior, su inicialización podría fallar y no intentar emitir ningún verbo LUA.

szDescription

Una cadena de texto que describe el software de LUA de Windows.

WinRUI

La aplicación utiliza esta función para emitir un verbo RUI. Si el verbo finaliza de manera asíncrona, LUA indicará la finalización emitiendo un mensaje al manejador de ventanas de la aplicación.

Antes de utilizar la llamada a WinRUI por primera vez, la aplicación debe utilizar RegisterWindowMessage para obtener el identificador de mensaje que LUA utilizará para los mensajes que indiquen una finalización asíncrona de verbo. Para ver más información, consulte “Uso” en la página 22.

Llamada de función

```
int WINAPI WinRUI (
    HWND hWnd,
    LUA_VERB_RECORD far * lpVCB
);
```

Para obtener la definición de la estructura LUA_VERB_RECORD, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Parámetros proporcionados

Los parámetros proporcionados son:

hWnd Un manejador de ventanas que LU utilizará para enviar un mensaje que indique una finalización asíncrona de verbo.

lpVCB Un puntero hacia la estructura de VCB para el verbo. Para la función WinRUI, el parámetro *lua_post_handle* está reservado; déjelo como 0 (cero).

Para obtener más información sobre la estructura de VCB, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47. Para obtener más información sobre su utilización para verbos individuales, consulte el Capítulo 4, “Verbos RUI”, en la página 59.

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice memset:

```
memset(vcb, 0, sizeof(vcb));
```

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

0 (cero)

La llamada de función se ha aceptado y el verbo LUA se procesará. La aplicación debería comprobar el parámetro *lua_flag2.async* de la estructura de VCB para determinar si el verbo ya ha finalizado de manera síncrona o

Puntos de entrada LUA para aplicaciones Windows

si finalizará de manera asíncrona, tal como se describe en el apartado “Finalización síncrona y asíncrona de verbos”.

WLUAINVALIDHANDLE

El parámetro *hWnd* no era un manejador de ventanas válido.

WLUASTARTUPNOTCALLED

La aplicación no ha emitido la llamada a *WinRUIStartup*, que es necesaria antes de emitir cualquier verbo LUA.

Para obtener información sobre los parámetros devueltos en la estructura de VCB, consulte las descripciones de los verbos individuales en el Capítulo 4, “Verbos RUI”, en la página 59.

Uso

Antes de utilizar *WinRUI* por primera vez, la aplicación debe utilizar la llamada a *RegisterWindowMessage* para obtener el identificador de mensaje que LUA utilizará para los mensajes que indiquen una finalización asíncrona de verbo.

RegisterWindowMessage es una llamada de función de Windows estándar, no específica de LUA; consulte la documentación de Windows para obtener más información sobre la función. (No es necesario emitir de nuevo la llamada antes de los verbos LUA subsiguientes; el valor devuelto será igual para todas las llamadas emitidas por la aplicación.)

La aplicación debe pasar la cadena *WinRUI* a la función; el valor devuelto es un identificador de mensaje (el valor devuelto desde la llamada a *RegisterWindowMessage*).

Cada vez que un verbo LUA que se ha emitido utilizando el punto de entrada *WinRUI* finaliza de manera asíncrona, LUA emite un mensaje al manejador de ventanas que se ha especificado en la llamada a *WinRUI*. El formato del mensaje es el siguiente:

- El identificador de mensaje es el valor que se devuelve de la llamada a *RegisterWindowMessage*.
- El argumento *lParam* contiene la dirección del VCB que se ha proporcionado a la llamada de *WinRUI* original; la aplicación puede utilizar esta dirección para acceder a los parámetros devueltos en la estructura de VCB.
- El argumento *wParam* no está definido.

Finalización síncrona y asíncrona de verbos

A veces LUA puede realizar todo el proceso de un verbo en cuanto se emite. En este caso, el verbo vuelve de manera síncrona; el código de retorno primario está definido en un valor distinto de *LUA_IN_PROGRESS*, y el bit *lua_flag2.async* tiene el valor 0 (cero). (Para ver información sobre estos parámetros devueltos, consulte el Capítulo 4, “Verbos RUI”, en la página 59.)

Para habilitar el verbo para que devuelva de manera asíncrona, la aplicación proporciona un manejador de ventanas al punto de entrada LUA. Si el verbo finaliza de manera síncrona, LUA no utiliza este manejador de ventanas. Si el verbo finaliza de manera asíncrona, LUA indica la finalización del verbo emitiendo un mensaje a este manejador de ventanas; este mensaje incluye un puntero hacia el VCB original.

Una aplicación no puede predecir si un determinado verbo finalizará de manera síncrona o asíncrona.

Los verbos se pueden emitir desde una rutina de devolución de llamada, pero no siempre finalizarán de manera asíncrona. Estos verbos se pueden devolver de manera síncrona si fallan desde la biblioteca. La aplicación no debería volver a emitir el verbo que ha fallado desde la rutina de devolución de llamada.

Si el usuario emite repetidamente RUI_INIT en paralelo desde el contexto de la rutina de devolución de llamada, a la larga los RUI_INIT fallarán con un error de memoria. Sin embargo, si se emiten verbos desde el subproceso de la aplicación, permitiendo la disponibilidad de toda la memoria del sistema, más intentos llegarán a finalizar satisfactoriamente.

WinRUIGetLastInitStatus

La aplicación utiliza esta función para determinar el estado de un verbo RUI_INIT anterior que todavía está pendiente. Puede utilizar la información que se devuelve para decidir si cancelará la iniciación de sesión (emitiendo RUI_TERM) o bien esperará a que se establezca la sesión.

La función se puede utilizar para realizar cualquiera de las acciones siguientes:

- Solicitar información sobre el estado actual de la sesión iniciada por un verbo RUI_INIT específico.
- Solicitar notificación asíncrona de los cambios producidos en el estado de sesión para una sesión específica o para todas las sesiones. Si el estado de sesión cambia, LUA lo indicará emitiendo un mensaje al manejador de ventanas de la aplicación o señalando el manejador de sucesos de la aplicación.
- Cancelar una petición anterior de notificación asíncrona de los cambios producidos en el estado de sesión.

Antes de utilizar la llamada a WinRUI por primera vez, la aplicación debe utilizar WinRUIStartup para registrarse como una aplicación LUA de Windows. Si necesita notificación asíncrona de los cambios de estado, también debe utilizar RegisterWindowMessage para obtener el identificador de mensaje que LUA utilizará para esta notificación. Para obtener más información sobre estas llamadas, consulte los apartados “WinRUIStartup” en la página 19 y “Uso” en la página 25.

Llamada de función

```
int WINAPI WinRUIGetLastInitStatus (  
    DWORD Sid,  
    HANDLE StatusHandle,  
    DWORD NotifyType,  
    BOOL ClearPrevious  
);
```

Parámetros proporcionados

Los parámetros proporcionados son:

Sid Para obtener información sobre el estado de sesión para un verbo RUI_INIT específico o para cancelar una petición previa de notificación de los cambios de estado de sesión para este verbo, especifique el ID de sesión que se devuelve inicialmente desde el verbo RUI_INIT.

Para solicitar notificación de los cambios de estado de sesión para todos los verbos RUI_INIT pendientes, especifique 0 (cero). En este caso, el parámetro *StatusHandle* debe especificar un manejador de Windows válido puesto que la información se devolverá siempre de manera asíncrona.

Para cancelar la notificación de los cambios de estado de sesión para todos los verbos RUI_INIT pendientes, especifique 0 (cero).

Puntos de entrada LUA para aplicaciones Windows

StatusHandle

Para obtener el estado de sesión actual para un verbo RUI_INIT específico, sin solicitar notificación de los cambios subsiguientes, especifique un manejador nulo.

Para solicitar notificación de los cambios de estado de sesión, para un verbo RUI_INIT específico o para todos los verbos RUI_INIT pendientes, especifique un manejador de Windows o un manejador de sucesos que LUA utilizará cuando cambie el estado de sesión.

Si el parámetro *ClearPrevious* está establecido en TRUE, para cancelar una petición de notificación anterior, LUA ignora este parámetro.

NotifyType

Si se solicita notificación asíncrona, este parámetro determina cómo LUA debe identificar el verbo RUI_INIT en el mensaje de notificación asíncrona. Los valores permitidos son:

WLUA_NTFY_MSG_CORRELATOR

El parámetro *StatusHandle* contiene un manejador de ventanas. Identifique el verbo utilizando el valor de *lua_correlator* proporcionado en el verbo RUI_INIT.

WLUA_NTFY_MSG_SID

El parámetro *StatusHandle* contiene un manejador de ventanas. Identifique el verbo utilizando el valor de *lua_sid* proporcionado en el verbo RUI_INIT.

WLUA_NTFY_EVENT

El parámetro *StatusHandle* contiene un manejador de sucesos.

Si el parámetro *StatusHandle* es nulo (para solicitar información de estado actual) o si el parámetro *ClearPrevious* está establecido en TRUE (para cancelar una petición de notificación anterior), LUA ignora este parámetro.

ClearPrevious

Para cancelar una petición de notificación anterior, establezca este parámetro en TRUE; LUA ignora los parámetros *StatusHandle* y *ClearPrevious*. Para solicitar el estado actual o la notificación de los cambios de estado futuros, establezca este parámetro en FALSE.

Valores devueltos

Si la función finaliza satisfactoriamente, el valor de retorno desde la función es uno de los siguientes:

WLUALINKINACTIVE

El enlace de comunicaciones con el sistema principal todavía no está activo.

WLUAUINACTIVE

El enlace de comunicaciones con el sistema principal está activo, pero todavía no se ha recibido un mensaje de activación de unidad física (ACTPU).

WLUAUACTIVE

Se ha recibido un mensaje ACTPU desde el sistema principal.

WLUAPUREACTIVATED

El sistema principal ha reactivado la PU.

Puntos de entrada LUA para aplicaciones Windows

WLUALUINACTIVE

El enlace de comunicaciones con el sistema principal está activo y se ha recibido un mensaje ACTPU, pero todavía no se ha recibido un mensaje ACTLU.

WLUALUACTIVE

La LU está activa.

WLUALUREACTIVATED

La LU se ha reactivado.

WLUAGETLU

La aplicación está estableciendo contacto con el nodo.

Si la aplicación ha solicitado notificación de los cambios de estado, uno de estos valores se incluirá en un mensaje de Windows que se enviará a la aplicación cada vez que cambie el estado. Para ver más información, consulte "Uso".

Los siguientes valores de retorno indican que la función ha fallado:

WLUASYSNOTREADY

El software SNA no está en ejecución.

WLUANTFYINVALID

El parámetro *NotifyType* se ha establecido en un valor que no es válido.

WLUAINVALIDHANDLE

El parámetro *StatusHandle* proporcionado no era un manejador de ventanas válido.

WLUASTARTUPNOTCALLED

La aplicación no ha emitido la llamada a *WinRUIStartup*, que es necesaria antes de emitir cualquier verbo LUA.

WLUAUNKNOWN

Error interno: se desconoce el estado de sesión.

WLUASIDINVALID

El parámetro *Sid* proporcionado no coincide con el ID de sesión de un verbo RUI_INIT pendiente.

WLUASIDZERO

La aplicación ha proporcionado un ID de sesión de cero (para indicar todas las sesiones), pero no ha especificado ningún manejador de Windows (para indicar notificación asíncrona) ni ningún valor de TRUE para *ClearPrevious* (para eliminar una petición de notificación anterior).

WLUAGLOBALHANDLER

La aplicación ha solicitado previamente la notificación de los cambios de estado para todos los verbos RUI_INIT; no puede solicitar notificación para una sesión específica a menos que primero elimine la notificación de "todas las sesiones".

Uso

Si la aplicación solicita notificación asíncrona de los cambios de estado utilizando un mensaje de Windows, debe utilizar la llamada a *RegisterWindowMessage* antes de su primera llamada a *WinRUIGetLastInitStatus*, para obtener el identificador de mensaje que LUA utilizará para los mensajes que indiquen cambios de estado.

La llamada a *RegisterWindowMessage* es una función de Windows estándar, no específica de LUA; consulte la documentación de Windows para obtener más

Puntos de entrada LUA para aplicaciones Windows

información sobre la función. (No es necesario emitir de nuevo la llamada antes de las llamadas subsiguientes a esta función; el valor devuelto será igual para todas las llamadas emitidas por la aplicación.)

La aplicación debe pasar la cadena "WinRUI" a la función; el valor devuelto es un identificador de mensaje (el valor devuelto desde la llamada a RegisterWindowMessage).

Cada vez que un estado de sesión cambia, LUA emite un mensaje al manejador de ventanas especificado en la llamada a WinRUI. El formato del mensaje es el siguiente:

- El identificador de mensaje es el valor que se devuelve de la llamada a RegisterWindowMessage.
- El argumento *lParam* contiene el valor de correlacionador proporcionado para el verbo RUI_INIT original o el ID de sesión devuelto en el verbo RUI_INIT original, tal como define el parámetro *NotifyType*. La aplicación puede utilizar este valor para correlacionar el mensaje con el verbo original.
- El argumento *wParam* contiene el estado de sesión (uno de los valores listados para ejecución satisfactoria en el apartado "Valores devueltos" en la página 24) o el valor WLUAUNKNOWN si se ha producido un error interno durante el proceso.

Si la aplicación solicita notificación asíncrona de los cambios de estado utilizando un manejador de sucesos, impleméntelo del modo siguiente:

```
WinRUIGetLastInitStatus(Sid,EventHandle,WLUA_NOTIFY_EVENT,FALSE);
```

El suceso del cual se proporciona el manejador se señalará cuando se produzca un cambio de estado. Puesto que no se devuelve información cuando se señala un suceso, debe emitirse una llamada adicional para determinar el estado, como se muestra a continuación:

```
Status = WinRUIGetLastInitStatus(Sid,NULL,0,FALSE);
```

Nota: En este caso, debe especificarse un *Sid*.

WinRUICleanup

La aplicación utiliza esta función para eliminar el registro como usuario de RUI de Windows, tras finalizar de emitir verbos RUI.

Llamada de función

```
BOOL WINAPI WinRUICleanup (void);
```

Parámetros proporcionados

No hay parámetros proporcionados para esta función.

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

- TRUE** Se ha eliminado el registro de la aplicación satisfactoriamente.
- FALSE** Se ha producido un error durante el proceso de la llamada y no se ha eliminado el registro de la aplicación. Compruebe los archivos de anotaciones cronológicas para ver si existen mensajes que indiquen la causa de la anomalía.

GetLuaReturnCode

La aplicación utiliza esta función para obtener una cadena de caracteres imprimibles que indique los códigos de retorno primario y secundario desde un VCB proporcionado. La cadena se puede utilizar para generar mensajes de error de aplicación para códigos de retorno non-LUA_OK.

Llamada de función

```
int WINAPI GetLuaReturnCode (
    struct LUA_COMMON FAR * vcbptr,
    unsigned int           buffer_length,
    unsigned char far *    buffer_addr
);
```

Parámetros proporcionados

Los parámetros proporcionados son:

vcbptr Un puntero hacia la estructura de VCB para el verbo. Para obtener más información sobre la estructura de VCB y sobre su utilización para verbos individuales, consulte el Capítulo 4, “Verbos RUI”, en la página 59.

buffer_length

La longitud (en bytes) del almacenamiento intermedio proporcionado por la aplicación para contener la cadena de datos que se devuelven. La longitud recomendada es 256 bytes.

buffer_addr

La dirección del almacenamiento intermedio proporcionado por la aplicación para contener la cadena de datos que se devuelven.

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

0 (cero)

La función ha finalizado satisfactoriamente.

0x20000001

LUA no ha podido leer desde el VCB suministrado o no ha podido grabar en el almacenamiento intermedio de datos suministrado.

0x20000002

El almacenamiento intermedio de datos suministrado es demasiado pequeño para contener la cadena de caracteres que se devuelve.

0x20000003

La biblioteca de enlace dinámico, **LUASTR32.DLL**, que genera las cadenas de caracteres que se devuelven para esta función, no se ha podido cargar.

Si el valor de retorno es 0 (cero), la cadena de caracteres que se devuelve se encuentra en el almacenamiento intermedio identificado por el parámetro *buffer_addr*. Esta cadena finaliza con un carácter de nulo (binario con ceros), pero no incluye un carácter (\n) de nueva línea final.

SLI

La aplicación utiliza esta función para emitir un verbo SLI LUA. Si el verbo finaliza de manera asíncrona, LUA indica la finalización señalando un manejador de sucesos proporcionado por la aplicación.

Llamada de función

```
void WINAPI SLI(verb)
LUA_VERB_RECORD FAR * verb;
```

Parámetros proporcionados

El parámetro proporcionado es:

verb Puntero a un bloque de control de verbos (VCB), que contiene los parámetros para que se emita el verbo. La estructura de VCB está definida en el archivo de cabecera de LUA **winlua.h**; este archivo se instala en el subdirectorio **/sdk** en el directorio en el que se ha instalado el software Windows Client. Para obtener una explicación sobre la estructura de VCB, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice `memset`:

```
memset(vcb, 0, sizeof(vcb));
```

Valores devueltos

El punto de entrada no devuelve ningún valor. Cuando la llamada vuelve, la aplicación puede examinar los parámetros del VCB para determinar si el verbo ha finalizado de manera síncrona o bien finalizará de manera asíncrona. Para ver más información, consulte “Uso”.

Uso

A veces LUA puede realizar todo el proceso de un verbo en cuanto se emite. En este caso, el verbo vuelve de manera síncrona; el código de retorno primario está definido en un valor distinto de `LUA_IN_PROGRESS`, y el bit `lua_flag2.async` tiene el valor 0 (cero). (Para ver información sobre estos parámetros devueltos, consulte el Capítulo 5, “Verbos SLI”, en la página 107.)

Otras veces, LUA debe esperar la información del nodo o la LU remota para poder completar el verbo. En tal caso, el verbo se devuelve de manera asíncrona; el código de retorno primario se establece en `LUA_IN_PROGRESS` y el bit `lua_flag2.async` es 1. Ahora la aplicación puede realizar otro proceso o bien esperar la notificación de LUA de que el verbo ha finalizado. LUA emite esta notificación estableciendo el código de retorno primario en su valor final, dejando el bit de `lua_flag2.async` en 1.

Como parte del VCB proporcionado, la aplicación suministra un manejador de sucesos en el parámetro `lua_post_handle`. El suceso debe estar en estado no señalado y el manejador debe tener acceso `EVENT_MODIFY_STATE` al suceso. Si el verbo finaliza de manera síncrona, LUA no señala este manejador de sucesos. Si el verbo finaliza de manera asíncrona, LUA indica la finalización del verbo señalando el manejador de sucesos.

Puntos de entrada LUA para aplicaciones Windows

La aplicación emite una llamada a `WaitForSingleObject` o `WaitForMultipleObject` para esperar el manejador de sucesos. Cuando se señala el suceso, la aplicación examina el código de retorno primario y el código de retorno secundario para buscar errores.

Una aplicación no puede predecir si un determinado verbo finalizará de manera síncrona o asíncrona.

WinSLIStartup

La aplicación utiliza esta función para registrarse como un usuario de SLI de Windows y para determinar si el software de LUA da soporte a la versión de LUA de Windows que necesita.

Llamada de función

```
int WINAPI WinSLIStartup (
    WORD wVersionRequired;
    LUADATA far * lpData;
)

typedef struct
{
    WORD wVersion;
    char szDescription[41];
} LUADATA;
```

Parámetros proporcionados

El parámetro proporcionado es:

wVersionRequired

La versión de LUA de Windows que la aplicación necesita. Communications Server para Linux soporta la Versión 1.0.

El byte de orden inferior especifica el número de versión más alto y el byte de orden superior especifica el número de versión más bajo. Por ejemplo:

| Versión | wVersionRequired |
|---------|------------------|
| 1.0 | 0x0001 |
| 1.1 | 0x0101 |
| 2.0 | 0x0002 |

Si la aplicación puede utilizar más de una versión, debe especificar la versión más alta que puede utilizar.

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

0 (cero)

La aplicación se ha registrado satisfactoriamente y el software de LUA de Windows da soporte al número de versión especificado por la aplicación o a una versión inferior. La aplicación debería comprobar el número de versión en la estructura LUADATA para asegurarse de que es lo suficientemente alto.

WLUAVERNOTSUPPORTED

El número de versión especificado por la aplicación no está soportado por el software de LUA de Windows. La aplicación no se ha registrado.

WLUAINITREJECT

La aplicación ya ha llamado a `WinSLIStartup` y se ha registrado satisfactoriamente. No debe llamarse a esta función más de una vez.

Puntos de entrada LUA para aplicaciones Windows

WLUASYSNOTREADY

El software Communications Server para Linux no se ha iniciado o el nodo local no está activo. La aplicación no se ha registrado.

WLUAFailure

Se ha producido un error del sistema operativo durante la inicialización del software de LUA de Windows. La aplicación no se ha registrado. Compruebe los archivos de anotaciones cronológicas para ver si existen mensajes que indiquen la causa de la anomalía.

Si el valor de retorno de `WinSLIStartup` es 0 (cero), la estructura `LUADATA` contiene información sobre el soporte proporcionado por el software de LUA de Windows. Si el valor de retorno es distinto de cero, el contenido de esta estructura no está definido y la aplicación no debería comprobarlo. Los parámetros de esta estructura son los siguientes:

wVersion

El número de versión de LUA de Windows al cual da soporte el software, en el mismo formato que el parámetro *wVersionRequired* (consulte "Parámetros proporcionados" en la página 19). Communications Server para Linux soporta la Versión 1.0.

Si el software da soporte al número de versión solicitado, este parámetro se establece en el mismo valor que el parámetro *wVersionRequired*; de lo contrario, se establece en la versión más alta a la que dé soporte el software, que será inferior al número de versión proporcionado por la aplicación. La aplicación debe comprobar el valor devuelto y realizar la acción que se indica a continuación:

- Si el número de versión devuelto es igual que el número de versión solicitado, la aplicación puede utilizar esta implementación de LUA de Windows.
- Si el número de versión devuelto es inferior al número de versión solicitado, la aplicación puede utilizar esta implementación de LUA de Windows pero no debe intentar utilizar características no soportadas por el número de versión devuelto. Si esto no es posible dado que necesita características que están disponibles en la versión inferior, su inicialización podría fallar y no intentar emitir ningún verbo LUA.

szDescription

Una cadena de texto que describe el software de LUA de Windows.

WinSLI

La aplicación utiliza esta función para emitir un verbo SLI. Si el verbo finaliza de manera asíncrona, LUA indicará la finalización emitiendo un mensaje al manejador de ventanas de la aplicación.

Antes de utilizar la llamada a `WinSLI` por primera vez, la aplicación debe utilizar `RegisterWindowMessage` para obtener el identificador de mensaje que LUA utilizará para los mensajes que indiquen una finalización asíncrona de verbo. Para ver más información, consulte "Uso" en la página 31.

Llamada de función

```
int WINAPI WinSLI (
    HWND hWnd,
    LUA_VERB_RECORD far * lpVCB
);
```

Para obtener la definición de la estructura `LUA_VERB_RECORD`, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Parámetros proporcionados

Los parámetros proporcionados son:

hWnd Un manejador de ventanas que LU utilizará para enviar un mensaje que indique una finalización asíncrona de verbo.

lpVCB Un puntero hacia la estructura de VCB para el verbo. Para la función `WinSLI`, el parámetro *lua_post_handle* está reservado; déjelo como 0 (cero).

Para obtener más información sobre la estructura de VCB, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47. Para obtener más información sobre su utilización para verbos individuales, consulte el Capítulo 5, “Verbos SLI”, en la página 107.

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice `memset`:

```
memset(vcb, 0, sizeof(vcb));
```

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

0 (cero)

La llamada de función se ha aceptado y el verbo LUA se procesará. La aplicación debería comprobar el parámetro *lua_flag2_async* de la estructura de VCB para determinar si el verbo ya ha finalizado de manera síncrona o si finalizará de manera asíncrona, tal como se describe en el apartado “Finalización síncrona y asíncrona de verbos” en la página 32.

WLUAINVALIDHANDLE

El parámetro *hWnd* no era un manejador de ventanas válido.

WLUASTARTUPNOTCALLED

La aplicación no ha emitido la llamada a `WinSLIStartup`, que es necesaria antes de emitir cualquier verbo SLI.

Para obtener información sobre los parámetros devueltos en la estructura de VCB, consulte las descripciones de los verbos individuales en el Capítulo 5, “Verbos SLI”, en la página 107.

Uso

Antes de utilizar `WinSLI` por primera vez, la aplicación debe utilizar la llamada a `RegisterWindowMessage` para obtener el identificador de mensaje que LUA utilizará

Puntos de entrada LUA para aplicaciones Windows

para los mensajes que indiquen una finalización asíncrona de verbo. `RegisterWindowMessage` es una llamada de función de Windows estándar, no específica de LUA; consulte la documentación de Windows para obtener más información sobre la función. (No es necesario emitir de nuevo la llamada antes de los verbos LUA subsiguientes; el valor devuelto será igual para todas las llamadas emitidas por la aplicación.)

La aplicación debe pasar la cadena `WinSLI` a la función; el valor devuelto es un identificador de mensaje (el valor devuelto desde la llamada a `RegisterWindowMessage`).

Cada vez que un verbo LUA que se ha emitido utilizando el punto de entrada `WinSLI` finaliza de manera asíncrona, LUA emite un mensaje al manejador de ventanas que se ha especificado en la llamada a `WinSLI`. El formato del mensaje es el siguiente:

- El identificador de mensaje es el valor que se devuelve de la llamada a `RegisterWindowMessage`.
- El argumento *lParam* contiene la dirección del VCB que se ha proporcionado a la llamada de `WinSLI` original; la aplicación puede utilizar esta dirección para acceder a los parámetros devueltos en la estructura de VCB.
- El argumento *wParam* no está definido.

Finalización síncrona y asíncrona de verbos

A veces LUA puede realizar todo el proceso de un verbo en cuanto se emite. En este caso, el verbo vuelve de manera síncrona; el código de retorno primario está definido en un valor distinto de `LUA_IN_PROGRESS`, y el bit `lua_flag2.async` tiene el valor 0 (cero). (Para ver información sobre estos parámetros devueltos, consulte el Capítulo 5, “Verbos SLI”, en la página 107.)

Para habilitar el verbo para que devuelva de manera asíncrona, la aplicación proporciona un manejador de ventanas al punto de entrada LUA. Si el verbo finaliza de manera síncrona, LUA no utiliza este manejador de ventanas. Si el verbo finaliza de manera asíncrona, LUA indica la finalización del verbo emitiendo un mensaje a este manejador de ventanas; este mensaje incluye un puntero hacia el VCB original.

Una aplicación no puede predecir si un determinado verbo finalizará de manera síncrona o asíncrona.

Los verbos se pueden emitir desde una rutina de devolución de llamada, pero no siempre finalizarán de manera asíncrona. Estos verbos se pueden devolver de manera síncrona si fallan desde la biblioteca. La aplicación no debería volver a emitir el verbo que ha fallado desde la rutina de devolución de llamada.

Si el usuario emite repetidamente `SLI_OPEN` en paralelo desde el contexto de la rutina de devolución de llamada, a la larga los `SLI_OPEN` fallarán con un error de memoria. Sin embargo, si se emiten verbos desde el subproceso de la aplicación, permitiendo la disponibilidad de toda la memoria del sistema, más intentos llegarán a finalizar satisfactoriamente.

WinSLICleanup

La aplicación utiliza esta función para eliminar el registro como usuario de SLI de Windows, tras finalizar de emitir verbos SLI.

Llamada de función

```
BOOL WINAPI WinSLICleanup (void);
```

Parámetros proporcionados

No hay parámetros proporcionados para esta función.

Valores devueltos

El valor de retorno desde la función es uno de los siguientes:

- TRUE** Se ha eliminado el registro de la aplicación satisfactoriamente.
- FALSE** Se ha producido un error durante el proceso de la llamada y la aplicación no se ha desregistrado. Compruebe los archivos de anotaciones cronológicas para ver si existen mensajes que indiquen la causa de la anomalía.



Emisión de un verbo LUA

Los pasos que deben llevarse a cabo para emitir un verbo LUA son los siguientes. Los ejemplos indican el uso del verbo RUI_INIT.

1. Incluya el archivo de cabecera LUA en el código fuente de la aplicación.

```
AIX, LINUX
```

```
#include < lua_c.h >
```

```
WINDOWS
```

```
#include < winlua.h >
```



```
AIX, LINUX
```

2. Defina una función de devolución de llamada que LUA utilizará para indicar que el verbo ha finalizado de manera asíncrona. (Para ver más información, consulte “Puntos de entrada LUA para aplicaciones AIX o Linux” en la página 13.)

```
void callback(verb)
LUA_VERB_RECORD * verb;
{
    .
    .
    .
}
```

```
WINDOWS
```

Si es el primer verbo LUA desde la aplicación y la aplicación va a emitir verbos RUI utilizando la llamada a WinRUI, emita la llamada a WinRUIStartup para inicializar la utilización de LUA de la aplicación. De modo similar, si la aplicación va a emitir verbos SLI utilizando la llamada a WinSLI, emita la llamada a WinSLIStartup para inicializar la utilización de LUA de la aplicación. (Para ver más información, consulte “Puntos de entrada LUA para aplicaciones

Emisión de un verbo LUA

Windows” en la página 16.) Esta llamada debe emitirse una vez antes del primer verbo LUA de la aplicación; no debe repetirse antes de los verbos subsiguientes.

Emita también la llamada a `RegisterWindowMessage` para obtener el identificador de mensaje que LUA utilizará cuando emita mensajes para indicar la finalización de un verbo LUA. (Para ver más información, consulte “Puntos de entrada LUA para aplicaciones Windows” en la página 16.) Esta llamada debe emitirse una vez antes del primer verbo LUA de la aplicación; no es necesario repetirla antes de los verbos subsiguientes.



3. Cree una variable para la estructura de VCB.

```
LUA_VERB_RECORD rui_init;
```

La estructura `LUA_VERB_RECORD` se declara en el archivo de cabecera `lua_c.h` (aplicaciones AIX o Linux) o `winlua.h` (aplicaciones Windows); para obtener una explicación de la estructura de VCB, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

4. Elimine (establezca a 0) las variables del VCB.

```
memset( rui_init, 0, sizeof( rui_init) );
```

LUA requiere que todos los parámetros reservados, y todos los parámetros no requeridos por el verbo que se está emitiendo, estén establecidos en 0 (cero). Para ver más información sobre los parámetros reservados, consulte “Formato de VCB (bloque de control de verbos) LUA” en la página 47. La manera más sencilla de hacerlo es establecer todo el VCB en ceros antes de definir los parámetros necesarios para este verbo en concreto.

5. Asigne valores a los parámetros del VCB que proporcionan información a LUA.

```
rui_init.common.lua_verb = LUA_VERB_RUI  
rui_init.common.lua_verb_length = sizeof(LUA_COMMON);  
rui_init.common.lua_opcode = LUA_OPCODE_RUI_INIT;  
memcpy( rui_init.common.lua_luname, "THISLU ", 8);
```

AIX, LINUX

```
rui_init.common.lua_post_handle = (unsigned long) callback;
```

WINDOWS

El parámetro `rui_init.common.lua_post_handle` está reservado; déjelo como 0 (cero).



Los valores `LUA_VERB_RUI` y `LUA_OPCODE_RUI_INIT` son constantes simbólicas. Estas constantes se definen en el archivo de cabecera `lua_c.h` (aplicaciones AIX o Linux) o `winlua.h` (aplicaciones Windows); se recomienda que utilice las constantes simbólicas y no los valores enteros, por portabilidad entre sistemas diferentes. (Para obtener más información, consulte el apartado “Desarrollo de aplicaciones portables” en la página 45.)

6. Invoque LUA. La dirección de la estructura de VCB es un parámetro de la llamada de función.

AIX, LINUX

```
RUI ( &rui_init );
```

WINDOWS

El punto de entrada WinRUI necesita un parámetro adicional, que es un manejador de ventanas para la ventana a la que LUA emitirá un mensaje para indicar la finalización asíncrona del verbo.

```
WinRUI ( handle, (LUA_VERB_RECORD far *) &rui_init );
```

7. Compruebe el parámetro *lua_flag2.async* para averiguar si el verbo ha finalizado de manera síncrona o finalizará de manera asíncrona.

```
if ( rui_init.common.lua_flag2.async )
{
    /* el verbo finalizará de manera asíncrona */
    /* con la rutina de devolución de llamada suministrada */
    /* continuar con otro proceso */
    .
    .
    .
}
else
{
    /* el verbo finaliza de manera síncrona */
    /* no se llamará a la rutina de devolución de llamada */
    /* procesar los valores devueltos aquí */
    .
    .
    .
}
```

Si el parámetro *lua_flag2.async* indica que el verbo finalizará de manera asíncrona, el subproceso de ejecución principal del programa no debería acceder a ningún otro parámetro del VCB en este punto. Cuando LUA llama a la rutina de devolución de llamada, la aplicación sí puede acceder a los parámetros del VCB.

8. Utilice las variables devueltas por LUA. Si el paso 7 indica que el verbo finalizará de forma asíncrona, este paso no se debe realizar hasta que se haya completado el verbo; en sistemas AIX o Linux, el proceso lo realiza normalmente la rutina de devolución de llamada. Si en el paso 7 se indica que el verbo ha finalizado de manera síncrona, el proceso debe llevarse a cabo mediante las instrucciones del código principal, porque no se llamará a la rutina de devolución de llamada.

```
if( rui_init.common.lua_prim_rc == LUA_OK )
{
    /* Inic OK */
    .
    .
    .
}
else
{
    /* Realizar rutina de error */
    .
    .
    .
}
```

Información de SNA

En este apartado se proporciona información sobre SNA que deberá tener en cuenta al escribir aplicaciones LUA de Communications Server para Linux para las comunicaciones con un sistema principal. Si está escribiendo una aplicación RUI primaria para las comunicaciones con una LU en sentido descendente, consulte el apartado “Información de SNA para RUI primaria” en la página 40.

Esta guía no pretende explicar detalladamente los conceptos de SNA. Si necesita información específica sobre los flujos de mensajes SNA, consulte la documentación de la aplicación de sistema principal para la que está diseñando la aplicación LUA de Communications Server para Linux.

Comprobación de BIND: RUI

Durante la inicialización de la sesión de LU, el sistema principal envía un mensaje BIND a la aplicación LUA de Communications Server para Linux que contiene información como, por ejemplo, los tamaños de RU que debe utilizar la sesión de LU. Communications Server para Linux devuelve este mensaje a la aplicación LUA en un verbo RUI_READ. La aplicación LUA debe comprobar que los parámetros especificados en el mensaje BIND son adecuados. Para ello, la aplicación dispone de las siguientes opciones:

- Aceptar el mensaje BIND tal cual, emitiendo un verbo RUI_WRITE que contenga una respuesta 0K a BIND. No es necesario enviar datos en la respuesta.
- Intentar negociar uno o varios parámetros BIND (sólo está permitido si BIND es negociable). Para ello, la aplicación emite un verbo RUI_WRITE que contiene una respuesta de tipo 0K, pero que incluye el BIND modificado como datos.
- Rechazar el mensaje BIND emitiendo un verbo RUI_WRITE que contiene una respuesta negativa, utilizando un código de detección SNA adecuado como datos.

Para ver más información sobre el verbo RUI_WRITE, consulte el Capítulo 4, “Verbos RUI”, en la página 59.

La validación de los parámetros BIND y la comprobación de la coherencia de todos los mensajes con dichos parámetros debe llevarla a cabo la aplicación LUA. No obstante, se aplican las dos restricciones siguientes:

- Communications Server para Linux rechaza cualquier verbo RUI_WRITE que especifique una longitud de RU mayor que el tamaño especificado en BIND.
- Communications Server para Linux requiere que BIND especifique que la LU secundaria es la ganadora de la contienda y que la recuperación de errores es responsabilidad de la perdedora de la contienda.

Comprobación de BIND: SLI

Durante la inicialización de la sesión de LU, el sistema principal envía un mensaje BIND a la aplicación LUA de Communications Server para Linux que contiene información como, por ejemplo, los tamaños de RU que debe utilizar la sesión de LU.

En el verbo SLI_OPEN, la aplicación puede especificar opcionalmente la dirección de su propia rutina para procesar peticiones de BIND desde el sistema principal. Si ya lo ha hecho, LUA envía un verbo SLI_BIND_ROUTINE adicional a la rutina proporcionada por la aplicación para permitir que procese la petición, tal como se

describe a continuación. La aplicación LUA debe comprobar que los parámetros especificados en el mensaje BIND son adecuados. Para ello, la aplicación dispone de las siguientes opciones:

- Aceptar el mensaje BIND tal cual, devolviendo el verbo SLI_BIND_ROUTINE con el código de retorno primario 0K. La aplicación no modifica el almacenamiento intermedio de datos que contiene el mensaje BIND.
- Intentar negociar uno o varios parámetros BIND (sólo está permitido si BIND es negociable). Para ello, la aplicación devuelve el verbo SLI_BIND_ROUTINE con el código de retorno primario 0K, pero que incluye el BIND modificado en el almacenamiento intermedio de datos.
- Rechazar el BIND devolviendo el verbo SLI_BIND_ROUTINE con el código de retorno primario LUA_NEGATIVE_RESPONSE y sustituyendo la petición de BIND del almacenamiento intermedio de datos por un código de detección SNA adecuado.

La validación de los parámetros BIND y la comprobación de la coherencia de todos los mensajes con dichos parámetros debe llevarla a cabo la aplicación LUA. Sin embargo, Communications Server para Linux requiere que BIND especifique que la LU secundaria es la ganadora de la contienda y que la recuperación de errores es responsabilidad de la perdedora de la contienda.

Respuestas negativas y códigos de detección SNA

Los códigos de detección SNA pueden devolverse a una aplicación LUA en los siguientes casos:

- Cuando el sistema principal envía una respuesta negativa a una petición de la aplicación LUA, ésta incluye un código de detección SNA que indica la razón de la respuesta negativa. Se informa de esto a la aplicación en un verbo RUI_READ o SLI_RECEIVE posterior, como se muestra a continuación:
 - El código de retorno primario es LUA_OK.
 - Los indicadores de petición/respuesta, tipo de respuesta y datos de detección incluidos tienen el valor 1, que indica una respuesta negativa que incluye datos de detección.
 - Los datos devueltos por el verbo RUI_READ o SLI_RECEIVE son el código de detección SNA.
- Cuando Communications Server para Linux recibe datos no válidos del sistema principal, generalmente envía una respuesta negativa al sistema principal y no pasa los datos que no son válidos a la aplicación LUA. Se informa de esto a la aplicación en un verbo RUI_READ o RUI_BID o SLI_RECEIVE / SLI_BID posterior, como se muestra a continuación:
 - El código de retorno primario es LUA_NEGATIVE_RSP.
 - El código de retorno secundario es el código de detección SNA enviado al sistema principal.
- En algunos casos, Communications Server para Linux detecta que los datos proporcionados por el sistema principal no son válidos, pero no puede determinar el código de detección correcto que se debe enviar. En este caso, pasa los datos que no son válidos en una petición de excepción (EXR) a la aplicación LUA en un verbo RUI_READ o SLI_RECEIVE, como se describe a continuación:
 - El indicador de petición/respuesta tiene el valor 0 (cero), que indica una petición.
 - El indicador de datos de detección incluidos tiene el valor 1, que indica que se incluyen datos de detección (este indicador normalmente se utiliza sólo para las peticiones).

Información de SNA

- Los datos del mensaje se sustituyen por un código de detección SNA sugerido.

A continuación, la aplicación debe enviar una respuesta negativa al mensaje; puede utilizar el código de detección sugerido por Communications Server para Linux o puede modificarlo.

- Communications Server para Linux puede enviar un código de detección a la aplicación para indicar que los datos proporcionados por la aplicación no eran válidos. Se informa de esto a la aplicación en el verbo RUI_WRITE o SLI_SEND que suministra los datos, como se indica a continuación:
 - El código de retorno primario es LUA_UNSUCCESSFUL.
 - El código de retorno secundario es el código de detección SNA.

Diferencia entre códigos de detección SNA y otros códigos de retorno secundarios

Nota: El orden de bytes utilizado en los códigos de retorno secundarios de LUA significa que el byte más significativo del valor numérico es el último byte, no el primer byte.

Para un código de retorno secundario que no es un código de detección, los dos bytes más significativos de este valor siempre son 0 (cero). Por ejemplo, 0x01000000 (LUA_INVALID_LUNAME) es un código de retorno secundario de LUA estándar y no un código de detección.

Para un código de detección SNA, los dos bytes más significativos no son cero; el byte más significativo proporciona la categoría de código de detección y el siguiente byte identifica un determinado código de detección dentro de dicha categoría. (Los bytes restantes pueden contener información adicional o pueden ser 0.) Por ejemplo, 0x00000108 (LUA_RESOURCE_NOT_AVAILABLE) es un código de detección.

Todos los códigos de retorno secundarios de LUA, incluidos los que son código de detección SNA, se listan en el Apéndice A, “Valores de código de retorno”, en la página 161.

Información de los códigos de detección SNA

Si necesita información sobre un código de detección devuelto, consulte la publicación *Systems Network Architecture: Formats* de IBM. Los códigos de detección aparecen en orden numérico por categoría.

También puede recuperar información de ayuda en línea acerca de un código de detección SNA específico generado en el sistema Communications Server para Linux, escribiendo **sna -getsense** seguido de la categoría y del modificador (los cuatro primeros dígitos) o el código de detección entero (los ocho dígitos) en la línea de mandatos. Para obtener más información, consulte la publicación *Communications Server para Linux, Guía de diagnósticos*.

Ritmo

El ritmo lo controla la interfaz de LUA; no es necesario que una aplicación LUA controle el ritmo, y dicha aplicación nunca debe activar el indicador de ritmo.

Si se utiliza el ritmo en los datos enviados desde la aplicación LUA al sistema principal (determinado por BIND), es posible que un verbo RUI_WRITE o

SLI_SEND tarde un poco en finalizar. Esto se debe a que, para poder enviar más datos, Communications Server para Linux tiene que esperar a recibir una respuesta de ritmo del sistema principal.

Si se utiliza una aplicación LUA para transferir grandes cantidades de datos en una dirección, ya sea a o desde el sistema principal (por ejemplo, una aplicación de transferencia de archivos), la configuración del sistema principal debe especificar que se utiliza el ritmo en esa dirección; así se garantiza que el nodo que recibe los datos no llega al límite de su capacidad de datos y que no se queda sin almacenamiento de datos.

Segmentación

La segmentación de las RU es controlada por la interfaz de LUA. LUA siempre pasa RU completas a la aplicación, y la aplicación debe pasar RU completas a LUA.

Modificación de bits RH (cabecera de petición/respuesta) del sistema principal no estándar

Un sistema principal puede enviar datos a una aplicación LUA con las opciones de inicio de corchete (BB) y de excepción de petición (RQE), pero sin la opción de fin de corchete (EB) (es decir, inicio de corchete y respuesta de excepción pero sin fin de corchete). Esta combinación de opciones no es estrictamente válida en SNA, pero la utilizan algunas aplicaciones de sistema principal.

Para soportar estas aplicaciones de sistema principal, Communications Server para Linux modifica los datos de sistema principal para especificar una respuesta precisa en lugar de una respuesta de excepción antes de enviarlos a la aplicación.

Acuses de recibo de cortesía

Communications Server para Linux mantiene un registro de las peticiones recibidas del sistema principal a fin de correlacionar las respuestas enviadas por la aplicación con la petición apropiada. Cuando la aplicación envía una respuesta, Communications Server para Linux la correlaciona con los datos de la petición original y, a continuación, puede liberar el almacenamiento asociado a ella.

Si el sistema principal especifica sólo una respuesta de excepción (se puede enviar una respuesta negativa pero no se debe enviar una respuesta afirmativa), Communications Server para Linux debe seguir manteniendo un registro de la petición por si la aplicación envía subsiguientemente una respuesta negativa. Si la aplicación no envía una respuesta, el almacenamiento asociado con esta petición no puede liberarse.

Debido a esto, Communications Server para Linux permite que la aplicación LUA emita una respuesta afirmativa a una petición de sólo respuesta de excepción del sistema principal (esto se conoce como acuse de recibo de cortesía). La respuesta no se envía al sistema principal, pero la utiliza Communications Server para Linux para borrar el almacenamiento asociado con la petición.

Eliminación de datos hasta el final de la cadena

Cuando el sistema principal envía una cadena de unidades de petición a una aplicación LUA, la aplicación puede esperar hasta que se reciba la última RU de la cadena antes de enviar una respuesta, o bien enviar una respuesta negativa a una RU que no sea la última de la cadena. Si se envía una respuesta negativa en mitad

Información de SNA

de la cadena, Communications Server para Linux depura todas las RU subsiguientes de esta cadena y no las envía a la aplicación.

Cuando Communications Server para Linux recibe la última RU de la cadena, lo indica a la aplicación estableciendo el código de retorno primario de un verbo RUI_READ o RUI_BID, o bien SLI_RECEIVE / SLI_BID, en LUA_NEGATIVE_RSP con un código de retorno secundario de 0 (cero).

El sistema principal puede finalizar la cadena enviando un mensaje como, por ejemplo, CANCEL en mitad de la cadena. En este caso, el mensaje CANCEL se devuelve a la aplicación en un verbo RUI_READ o SLI_RECEIVE y no se utiliza el código de retorno LUA_NEGATIVE_RSP (consulte "Respuestas negativas y códigos de detección SNA" en la página 37).

Información de SNA para RUI primaria

En este apartado se proporciona información sobre SNA que necesita tener en cuenta al escribir aplicaciones RUI primarias de Communications Server para Linux para las comunicaciones con una LU en sentido descendente.

Esta guía no pretende explicar detalladamente los conceptos de SNA. Si necesita información específica sobre los flujos de mensajes SNA, consulte la documentación de la aplicación de sistema principal para la que está diseñando la aplicación LUA de Communications Server para Linux.

Responsabilidades de la aplicación RUI primaria

Una aplicación RUI primaria tiene control de las sesiones de LU-SSCP y PLU-SLU a nivel de Unidad de petición/respuesta (RU) y puede enviar y recibir las RU SNA en estas sesiones. La sesión de PU-SSCP es interna de Communications Server para Linux y la aplicación RUI primaria no puede acceder a ella.

Dado que una aplicación RUI primaria funciona a nivel de RU, tiene un alto grado de control sobre el flujo de datos destinado y procedente de la LU secundaria. Sin embargo, tiene una responsabilidad mayor que una aplicación LUA normal para asegurar que los mensajes SNA que envía son válidos y que los protocolos de nivel de RU (por ejemplo delimitadores y encadenamiento) se utilizan correctamente. En particular, tenga en cuenta que Communications Server para Linux no intenta verificar la validez de las RU enviadas por una aplicación RUI primaria.

La aplicación RUI primaria es responsable de:

- Inicializar las LU en sentido descendente utilizando RUI_INIT_PRIMARY y terminándola utilizando RUI_TERM
- Procesar mensajes NOTIFY de la LU secundaria mientras las aplicaciones secundarias se inician y se detienen
- Procesar INIT-SELF y TERM-SELF para activar y desactivar la sesión de PLU-SLU
- Crear, enviar, recibir y analizar mensajes de corrientes de datos 3270 en las RU de datos
- Implementar protocolos a nivel de RU (control de petición, delimitadores, encadenamiento, dirección)
- Criptografía (si es necesaria)
- Compresión (si es necesaria).

Ritmo

El ritmo lo controla la interfaz de LUA; no es necesario que una aplicación LUA controle el ritmo, y dicha aplicación nunca debe activar el indicador de ritmo.

Si se está utilizando ritmo en los datos enviados desde la aplicación LUA al sistema principal (esto lo determina BIND), es posible que un verbo RUI_WRITE tarde cierto tiempo en completarse. Esto se debe a que, para poder enviar más datos, Communications Server para Linux tiene que esperar a recibir una respuesta de ritmo del sistema principal.

Si se utiliza una aplicación LUA para transferir grandes cantidades de datos en una dirección, ya sea a o desde el sistema principal (por ejemplo, una aplicación de transferencia de archivos), la configuración del sistema principal debe especificar que se utiliza el ritmo en esa dirección; así se garantiza que el nodo que recibe los datos no llega al límite de su capacidad de datos y que no se queda sin almacenamiento de datos.

Segmentación

La segmentación de las RU es controlada por la interfaz de LUA. LUA siempre pasa RU completas a la aplicación, y la aplicación debe pasar RU completas a LUA.

Restricciones

Communications Server para Linux no soporta lo siguiente para las aplicaciones RUI primarias:

- PU en sentido descendente a través de DLUR
- LU dependientes definidas dinámicamente (DDDLU)
- Envío de STSN (para restablecer números de secuencia, la aplicación debe ejecutar UNBIND y volver a ejecutar BIND para la sesión).

Acuses de recibo de cortesía

Communications Server para Linux mantiene un registro de las peticiones recibidas del sistema principal a fin de correlacionar las respuestas enviadas por la aplicación con la petición apropiada. Cuando la aplicación envía una respuesta, Communications Server para Linux la correlaciona con los datos de la petición original y, a continuación, puede liberar el almacenamiento asociado a ella.

Si el sistema principal especifica sólo una respuesta de excepción (se puede enviar una respuesta negativa pero no se debe enviar una respuesta afirmativa), Communications Server para Linux debe seguir manteniendo un registro de la petición por si la aplicación envía subsiguientemente una respuesta negativa. Si la aplicación no envía una respuesta, el almacenamiento asociado con esta petición no puede liberarse.

Debido a esto, Communications Server para Linux permite que la aplicación LUA emita una respuesta afirmativa a una petición de sólo respuesta de excepción del sistema principal (esto se conoce como acuse de recibo de cortesía). La respuesta no se envía al sistema principal, pero la utiliza Communications Server para Linux para borrar el almacenamiento asociado con la petición.

Eliminación de datos hasta el final de la cadena

Cuando el sistema principal envía una cadena de unidades de petición a una aplicación LUA, la aplicación puede esperar hasta que se reciba la última RU de la

Información de SNA para RUI primaria

cadena antes de enviar una respuesta, o bien enviar una respuesta negativa a una RU que no sea la última de la cadena. Si se envía una respuesta negativa en mitad de la cadena, Communications Server para Linux depura todas las RU subsiguientes de esta cadena y no las envía a la aplicación.

Cuando Communications Server para Linux recibe la última RU de la cadena, lo indica a la aplicación estableciendo el código de retorno primario de un verbo RUI_READ o RUI_BID en LUA_NEGATIVE_RSP con un código de retorno secundario de 0 (cero).

El sistema principal puede finalizar la cadena enviando un mensaje como, por ejemplo, CANCEL en mitad de la cadena. En este caso, se devuelve el mensaje CANCEL a la aplicación en un verbo RUI_READ y no se utiliza el código de retorno LUA_NEGATIVE_RSP (consulte el apartado “Respuestas negativas y códigos de detección SNA” en la página 37).

Información de configuración

El archivo de configuración de Communications Server para Linux, configurado y mantenido por el Administrador del sistema, contiene información necesaria para que las aplicaciones LUA se comuniquen. Para obtener información adicional, consulte la publicación *Communications Server para Linux, Guía de administración*.

AIX, LINUX

Para una aplicación RUI primaria que se comunica con una LU en sentido descendente, la única configuración necesaria es la LU en sentido descendente (o una plantilla de PU en sentido descendente).



Se deben configurar los componentes siguientes para utilizarlos con una aplicación LUA que se comunique con un sistema principal:

Control de enlace de datos (DLC), puerto y estación de enlace (LS)

Componentes de comunicaciones que Communications Server para Linux utiliza para comunicarse con el sistema principal remoto.

LU

Una LU de tipo 0–3, con un número de LU que coincide con el de una LU adecuada en el sistema principal.

Agrupación de LU (opcional)

Si es necesario, puede configurar más de una LU para que las utilice la aplicación y agrupar dichas LU en una agrupación. Esto significa que una aplicación puede especificar la agrupación en lugar de una LU específica al intentar iniciar una sesión y se le asignará la LU utilizada menos recientemente de la agrupación.

Una aplicación LUA indica a Communications Server para Linux que desea iniciar una sesión emitiendo un verbo RUI_INIT o SLI_OPEN con un nombre de LU. Este

nombre debe coincidir con el nombre de una LU de tipo 0-3, o de una agrupación de LU del archivo de configuración. Communications Server para Linux utiliza este nombre del modo siguiente:

- Si el nombre proporcionado es el nombre de una LU que no se encuentra en una agrupación, se asignará una sesión utilizando dicha LU si está disponible (es decir, si todavía no la utiliza ningún programa).
- Si el nombre proporcionado es el nombre de una agrupación de LU o el nombre de cualquier LU de la agrupación, se asignará una sesión utilizando la LU mencionada si está disponible o, de lo contrario, la LU utilizada menos recientemente de la agrupación. El verbo RUI_INIT o SLI_OPEN devuelve el nombre de la LU real asignada (que no puede ser igual que el nombre especificado). La aplicación puede utilizar este nombre de LU devuelto en verbos LUA posteriores para identificar la sesión.

Consideraciones acerca de AIX o Linux

AIX, LINUX

Este apartado resume las consideraciones de proceso que debe tener en cuenta al desarrollar aplicaciones LUA en un sistema AIX o Linux.

Archivo de cabecera LUA

El archivo de cabecera a utilizar con aplicaciones LUA es **lua_c.h**. Este archivo contiene las definiciones de los puntos de entrada LUA y los VCB LUA. También incluye el archivo de cabecera de interfaz común **values_c.h**; estos dos archivos contienen todas las constantes definidas para los valores de parámetros proporcionados y devueltos en la interfaz de LUA. El archivo **values_c.h** también incluye las definiciones de tipos de parámetro, como AP_UINT16, que se utilizan en los VCB LUA.

Estos dos archivos se almacenan en **/usr/include/sna** (AIX) o en **/opt/ibm/sna/include** (Linux).

Varios procesos y varias sesiones

Si el proceso que ha emitido RUI_INIT, RUI_INIT_PRIMARY o SLI_OPEN se bifurca para crear un proceso hijo, el proceso hijo no puede emitir ningún verbo LUA en la sesión iniciada por el proceso padre; los verbos fallarán y devolverán los códigos de retorno LUA_UNSUCCESSFUL y LUA_INVALID_PROCESS. Sin embargo, puede emitir otro RUI_INIT, RUI_INIT_PRIMARY o SLI_OPEN para obtener su propia sesión.

Un solo proceso puede utilizar simultáneamente más de una sesión LUA, emitiendo varios verbos RUI_INIT, RUI_INIT_PRIMARY o SLI_OPEN. Cada sesión debe utilizar una LU diferente, pero dos o más sesiones pueden utilizar la misma agrupación.

Dos o más instancias de la misma aplicación LUA pueden ejecutarse como procesos diferentes, pero deben utilizar LU diferentes. Para ello, debe proporcionarse un mecanismo para especificar el nombre de la LU durante la ejecución o utilizar las agrupaciones de LU; si los dos procesos especifican la misma agrupación, se les asignarán diferentes LU de dicha agrupación.

Compilación y enlace de la aplicación LUA

Aplicaciones AIX

Para compilar y enlazar aplicaciones de 32 bits, utilice las opciones siguientes:

```
-bimport:/usr/lib/sna/lua_r.exp -I  
/usr/include/sna
```

Para compilar y enlazar aplicaciones de 64 bits, utilice las opciones siguientes:

```
-bimport:/usr/lib/sna/lua_r64_5.exp -I  
/usr/include/sna
```

Aplicaciones Linux

Antes de compilar y enlazar una aplicación LUA, especifique el directorio en el que se almacenan las bibliotecas compartidas, de modo que la aplicación pueda encontrarlas durante el tiempo de ejecución. Para ello, establezca la variable de entorno LD_RUN_PATH en **/opt/ibm/sna/lib** o en **/opt/ibm/sna/lib64** si está compilando una aplicación de 64 bits.

Para compilar y enlazar aplicaciones de 32 bits, utilice las opciones siguientes:

```
-I /opt/ibm/sna/include -L  
/opt/ibm/sna/lib -llua -lsna_r -lpthread -lpLis
```

Para compilar y enlazar aplicaciones de 64 bits, utilice las opciones siguientes:

```
-I /opt/ibm/sna/include -L  
/opt/ibm/sna/lib64 -llua -lsna_r -lpthread -lpLis
```

Consideraciones acerca de Windows

WINDOWS

En este apartado se resumen las consideraciones de proceso que deben tenerse en cuenta cuando se desarrollan aplicaciones LUA en un cliente Windows.

Múltiples sesiones y múltiples tareas

Una sola tarea puede utilizar simultáneamente más de una sesión LUA, emitiendo varios verbos RUI_INIT o SLI_OPEN. Cada sesión debe utilizar una LU diferente, pero dos o más sesiones pueden utilizar la misma agrupación.

Dos o más instancias de la misma aplicación LUA pueden ejecutarse como tareas diferentes, pero deben utilizar LU diferentes. Para ello se utilizan agrupaciones de LU; las dos tareas pueden especificar la misma agrupación, pero se les asignarán LU diferentes de dicha agrupación.

Compilación y enlace de programas LUA

Este apartado proporciona información sobre cómo compilar y enlazar programas LUA en un cliente Windows.

Opciones de compilador para empaquetado de estructuras

Las estructuras del VCB para verbos LUA no están empaquetadas. No utilice opciones de compilador que cambien este método de empaquetado.

Los parámetros *DWORD* están dentro de los límites de *DWORD*, los parámetros *WORD* están dentro de los límites de *WORD* y los parámetros *BYTE* están dentro de los límites de *BYTE*.

Archivo de cabecera

El archivo de cabecera que se debe incluir en las aplicaciones LUA de Windows se denomina **winlua.h**. Este archivo está instalado en el subdirectorio `\sdk` para aplicaciones de 32 bits o `\sdk64` para aplicaciones de 64 bits, en el directorio donde ha instalado el software Remote API Client en Windows.

Enlace en tiempo de carga

Para enlazar el programa a LUA durante la carga, enlace el programa a la biblioteca **winsli32.lib** (para SLI) o la biblioteca **winrui32.lib** (para RUI). Este archivo se almacena en el subdirectorio `\sdk` para aplicaciones de 32 bits o `\sdk64` para aplicaciones de 64 bits,

en el directorio donde ha instalado el software de cliente de Windows

Enlace en tiempo de ejecución

Para enlazar el programa con LUA durante el tiempo de ejecución, incluya las llamadas siguientes en el programa:

- `LoadLibrary` para cargar la biblioteca de enlaces dinámicos de LUA **winsli32.dll** (para SLI) o **winrui32.dll** (para RUI).
- `GetProcAddress` para especificar cada uno de los puntos de entrada LUA necesarios (tales como SLI)
- `FreeLibrary` cuando la biblioteca ya no es necesaria.

Finalización de aplicaciones

Si una aplicación debe cerrarse (por ejemplo, si recibe un mensaje `WM_CLOSE` como resultado de `ALT F4`), deberá llamar a la función `WinRUICleanup` o `WinSLICleanup` antes de que un usuario haya pulsado finalizar. Si no lo hace, la aplicación queda en un estado indeterminado, a pesar de que se lleva a cabo tanta limpieza como es posible cuando el software de LUA de Windows detecta que la aplicación ha finalizado.



Desarrollo de aplicaciones portables

La implementación de Communications Server para Linux de LUA está diseñada para ser compatible con la implementación proporcionada por OS/2 Extended Edition de IBM. No obstante, hay algunas diferencias entre las implementaciones, debido a diferencias esenciales de los sistemas operativos. Estas diferencias de los sistemas operativos están indicadas en las descripciones de los verbos. Concretamente, son:

- El verbo `RUI_REINIT` es una ampliación de la especificación de la interfaz de LUA estándar. No está disponible en Remote API Client en Windows y puede no estar disponible en otras implementaciones de LUA.

Desarrollo de aplicaciones portables

- Otras implementaciones de LUA generan determinados códigos de retorno adicionales no devueltos por la implementación de Communications Server para Linux; es posible que también utilicen parámetros que están reservados para Communications Server para Linux.
- Las implementaciones de OS/2 y Windows utilizan puntos far (far *) en todos los casos; las implementaciones de AIX o Linux no tienen el concepto de punteros far y near, de modo que la palabra far se debe omitir para las implementaciones de AIX o Linux.
- La función de retorno asíncrono de verbos está soportada de diferente manera en los diferentes sistemas operativos. Puede que deba volver a escribir las secciones de una aplicación LUA desarrollada para un sistema operativo relacionadas con verbos devueltos de forma asíncrona si está portando la aplicación a otro sistema operativo.
- Puede que otras implementaciones de LUA no soporten agrupaciones de LU.

Se proporcionan las directrices siguientes para escribir aplicaciones LUA de Communications Server para Linux que sean portables a otros entornos:

- Incluya el archivo de cabecera de LUA sin ningún prefijo de nombre de vía de acceso. Esto permite utilizar la aplicación en un entorno que tenga otro sistema de archivos. Utilice opciones de inclusión en el compilador para localizar el archivo (consulte “Compilación y enlace de la aplicación LUA” en la página 44).
- Utilice los nombres de constantes simbólicas para los valores de parámetro y códigos de retorno, no los valores numéricos mostrados en el archivo de cabecera; esto garantiza que se utilizará el valor correcto, independientemente de cómo estén almacenados estos valores en la memoria.
- Cuando acceda a los códigos de detección SNA en un almacenamiento intermedio de datos, utilice las constantes simbólicas en lugar de los valores numéricos; esto garantiza que el orden de almacenamiento de bytes será correcto para el sistema.
- Incluya una comprobación para saber si hay los códigos de retorno que no sean aplicables al sistema operativo actual (por ejemplo, utilizando un caso “default” en una sentencia switch) y proporcione los diagnósticos adecuados.
- Compruebe que los parámetros mostrados como reservados tienen el valor 0 (cero).
- Establezca el parámetro *lua_verb_length* tal como se describe en el Capítulo 4, “Verbos RUI”, en la página 59 o el Capítulo 5, “Verbos SLI”, en la página 107.

Capítulo 3. Estructura de VCB LUA

Este capítulo contiene detalles de la estructura del bloque de control de verbos LUA que se utiliza para todos los verbos LUA.

Las constantes simbólicas se definen en los archivos de cabecera `lua_c.h` y `values_c.h` (sistema operativo AIX o Linux) o `winlua.h` (sistema operativo Windows) para muchos valores de parámetros. Por motivos de portabilidad, utilice la constante simbólica y no el valor numérico cuando establezca valores para los parámetros proporcionados o cuando pruebe los valores de los parámetros devueltos. El archivo `values_c.h` también incluye las definiciones de tipos de parámetro, como `AP_UINT16`, que se utilizan en los VCB LUA.

Los parámetros marcados como “reservados” siempre deben tener el valor 0 (cero).

Formato de VCB (bloque de control de verbos) LUA

El bloque de control de verbos consta de dos partes:

- Una estructura de datos **común**, utilizada para todos los verbos.
- Una estructura de datos **específica**, que se utiliza únicamente para los verbos siguientes:
 - `RUI_BID`
 - La versión ampliada de `RUI_INIT` (en el entorno AIX o Linux)
 - `SLI_BID`
 - `SLI_OPEN`
 - `SLI_SEND`

La definición de algunas partes de la estructura de VCB, en concreto el orden de los campos de bit, varía en los diferentes sistemas operativos. Para mayor claridad, aquí sólo se muestra una versión del orden, aunque en el archivo de cabecera se definen las dos versiones. Al establecer o probar valores en los campos de bit, la aplicación debe acceder a los bits individuales por nombre, a fin de evitar dependencias del orden de los bits, en lugar de utilizar las operaciones AND u OR a nivel de bit en bytes completos.

AIX, LINUX

Para permitir estas diferencias, el archivo de cabecera LUA contiene la siguiente información:

- Una sentencia `#include` para el archivo `/usr/include/sna/svconfig.h` (AIX) o `/opt/ibm/sna/include/svconfig.h` (Linux).
- La definición de tipo para los campos de bit en las estructuras de datos LUA. Esta definición garantiza que las estructuras de datos se almacenan en el formato correcto. La definición depende del valor de `PUCHARQD`, que está en el archivo `svconfig.h`.

Formato de VCB (bloque de control de verbos) LUA

Nota: El VCB LUA contiene muchos parámetros marcados como “reservados”; el software Communications Server para Linux utiliza internamente algunos de estos parámetros y otros no se utilizan en esta versión pero es posible que se utilicen en versiones futuras. Su aplicación no debe intentar acceder a los parámetros reservados; en cambio, debe definir todo el contenido del VCB en cero para garantizar que todos estos parámetros sean cero, antes de definir otros parámetros utilizados por el verbo. Esto asegura que Communications Server para Linux no interprete incorrectamente ninguno de los parámetros utilizados internamente y también que la aplicación continúe funcionando con las versiones futuras de Communications Server para Linux en las que estos parámetros se puedan utilizar para proporcionar funciones nuevas.

Para definir el contenido del VCB en cero, utilice memset:

```
memset(vcb, 0, sizeof(vcb));
```

Estructura de datos LUA_VERB_RECORD

```
typedef struct
{
    struct LUA_COMMON common;
    struct LUA_SPECIFIC specific;
} LUA_VERB_RECORD;
```

Estructura de datos común

AIX, LINUX

```
struct LUA_COMMON
{
    AP_UINT16    lua_verb;                /* Código de verbo */
    AP_UINT16    lua_verb_length;        /* Longitud registro de verbo */
    AP_UINT16    lua_prim_rc;            /* Código de retorno primario */
    AP_UINT32    lua_sec_rc;             /* Código retorno secundario */
    AP_UINT16    lua_opcode;             /* Código operación de verbo */
    AP_UINT32    lua_correlator;         /* Campo correlación usuario */
    unsigned char lua_luname[8];         /* Nombre de LU local */
    AP_UINT16    lua_extension_list_offset; /* Desplaz. lista exten. DLL */
    AP_UINT16    lua_cobol_offset;       /* Desplaz. extensión Cobol */
    AP_UINT32    lua_sid;                /* Identificador de sesión */
    AP_UINT16    lua_max_length;         /* Long. alm. intermedio rec. */
    AP_UINT16    lua_data_length;        /* Longitud de datos */
    char *       lua_data_ptr;           /* Puntero alm. interm. datos */
    unsigned long lua_post_handle;        /* Manejador de envío */

    struct LUA_TH {                      /* Campos TH LUA */
        BIT_FIELD_TYPE flags_fid : 4;    /* Identificac. formato tipo 2 */
        BIT_FIELD_TYPE flags_mpf : 2;    /* Campo correlación segmentac. */
        BIT_FIELD_TYPE flags_odai : 1;   /* Indicador asignador OAF-DAF */
        BIT_FIELD_TYPE flags_efi : 1;    /* Indicador de flujo rápido */
        BIT_FIELD_TYPE          : 8;    /* Campo reservado */
        unsigned char daf;               /* Campo dirección destino */
        unsigned char oaf;               /* Campo dirección origen */
        unsigned char snf[2];            /* Campo de número de secuencia */
    } lua_th;

    struct LUA_RH {                      /* Campos RH LUA */
        BIT_FIELD_TYPE rri : 1;          /* Indicador petición/respuesta */
        BIT_FIELD_TYPE ruc : 2;          /* Categoría RU */
        BIT_FIELD_TYPE          : 1;    /* Campo reservado */
        BIT_FIELD_TYPE fi : 1;           /* Indicador de formato */
        BIT_FIELD_TYPE sdi : 1;          /* Ind. datos detec. incluidos */
        BIT_FIELD_TYPE bci : 1;          /* Indicador inicio de cadena */
    } lua_rh;
};
```

Formato de VCB (bloque de control de verbos) LUA

```

BIT_FIELD_TYPE  eci : 1;          /* Indicador fin de cadena */

BIT_FIELD_TYPE  dr1i : 1;        /* Indicador DR 1 */
BIT_FIELD_TYPE  : 1;          /* Campo reservado */
BIT_FIELD_TYPE  dr2i : 1;        /* Indicador DR 2 */
BIT_FIELD_TYPE  ri : 1;         /* Indicador de respuesta */
BIT_FIELD_TYPE  : 2;          /* Campo reservado */
BIT_FIELD_TYPE  qri : 1;        /* Indicador respuesta en cola */
BIT_FIELD_TYPE  pi : 1;         /* Indicador de ritmo */

BIT_FIELD_TYPE  bbi : 1;        /* Indicador inicio de corchete */
BIT_FIELD_TYPE  ebi : 1;        /* Indicador fin de corchete */
BIT_FIELD_TYPE  cdi : 1;        /* Indicador cambio dirección */
BIT_FIELD_TYPE  : 1;          /* Campo reservado */
BIT_FIELD_TYPE  csi : 1;        /* Indicador selección código */
BIT_FIELD_TYPE  edi : 1;        /* Indicador datos cifrados */
BIT_FIELD_TYPE  pdi : 1;        /* Indicador datos rellenos */
BIT_FIELD_TYPE  : 1;          /* Campo reservado */
} lua_rh;

struct LUA_FLAG1 {
BIT_FIELD_TYPE  bid_enable : 1;  /* Indicador habilitado bid */
BIT_FIELD_TYPE  reserv1 : 1;    /* Reservado */
BIT_FIELD_TYPE  close_abend : 1; /* Indicador cierre inmediato */
BIT_FIELD_TYPE  nowait : 1;     /* Indicador no esperar datos */
BIT_FIELD_TYPE  sscp_exp : 1;   /* Flujo rápido de SSCP */
BIT_FIELD_TYPE  sscp_norm : 1;  /* Flujo normal de SSCP */
BIT_FIELD_TYPE  lu_exp : 1;     /* Flujo rápido de LU */
BIT_FIELD_TYPE  lu_norm : 1;    /* Flujo normal de LU */
} lua_flag1;

unsigned char   lua_message_type; /* Tipo mandato mensaje SNA */

struct LUA_FLAG2 {
BIT_FIELD_TYPE  bid_enable : 1;  /* Indicador habilitado bid */
BIT_FIELD_TYPE  async : 1;      /* Indicador finalización
asíncrona de verbo */
BIT_FIELD_TYPE  : 2;          /* Reservado */
BIT_FIELD_TYPE  sscp_exp : 1;   /* Flujo rápido de SSCP */
BIT_FIELD_TYPE  sscp_norm : 1;  /* Flujo normal de SSCP */
BIT_FIELD_TYPE  lu_exp : 1;     /* Flujo rápido de LU */
BIT_FIELD_TYPE  lu_norm : 1;    /* Flujo normal de LU */
} lua_flag2;

unsigned char   lua_resv56[7];    /* Campo reservado */
unsigned char   lua_encr_decr_option; /* Opción de criptografía */
} ;

```

WINDOWS

```

struct LUA_COMMON
{
unsigned short  lua_verb;          /* Código de verbo */
unsigned short  lua_verb_length;  /* Long. de registro de verbo */
unsigned short  lua_prim_rc;      /* Código de retorno primario */
unsigned long   lua_sec_rc;       /* Código de retorno secundario*/
unsigned short  lua_opcode;       /* Código de operación de verbo*/
unsigned long   lua_correlator;   /* Campo de correlación usuario*/
unsigned char   lua_luname[8];   /* Nombre de LU local */
unsigned short  lua_extension_list_offset; /* Desplaz. lista ampliacion DLL */
unsigned short  lua_cobol_offset; /* Desplaz. de ampliación Cobol*/
unsigned long   lua_sid;         /* ID de sesión */
unsigned short  lua_max_length;  /* Long. alm. interm. recepción*/
unsigned short  lua_data_length; /* Longitud de datos */
char far       *lua_data_ptr;    /* Puntero alm. interm. datos */
unsigned long   lua_post_handle; /* Manejador de envío */
}

```

Formato de VCB (bloque de control de verbos) LUA

```

struct LUA_TH {
    unsigned char    flags_fid : 4;    /* Campos TH LUA */
    unsigned char    flags_mpf : 2;    /* Identificac. formato tipo 2 */
    unsigned char    flags_odai : 1;   /* Campo correlación segmentac. */
    unsigned char    flags_efi : 1;   /* Indicador asignador OAF-DAF */
    unsigned char    : 8;             /* Indicador de flujo rápido */
    unsigned char    daf;             /* Campo reservado */
    unsigned char    oaf;             /* Campo dirección destino */
    unsigned char    snf[2];          /* Campo dirección origen */
    } lua_th;                          /* Campo de número de secuencia */

struct LUA_RH {
    unsigned char    rri : 1;         /* Campos RH LUA */
    unsigned char    ruc : 2;         /* Indicador petición/respuesta */
    unsigned char    : 1;            /* Categoría RU */
    unsigned char    fi : 1;          /* Campo reservado */
    unsigned char    sdi : 1;         /* Indicador de formato */
    unsigned char    bci : 1;         /* Ind. datos detec. incluidos */
    unsigned char    eci : 1;         /* Ind. datos detec. incluidos */
    unsigned char    : 1;            /* Indicador de inicio de cadena */
    unsigned char    dr1i : 1;        /* Indicador de fin de cadena */
    unsigned char    : 1;            /* Indicador DR 1 */
    unsigned char    dr2i : 1;        /* Campo reservado */
    unsigned char    ri : 1;          /* Indicador DR 2 */
    unsigned char    : 2;            /* Indicador de respuesta */
    unsigned char    qri : 1;         /* Campo reservado */
    unsigned char    pi : 1;          /* Indicador respuesta en cola */
    unsigned char    : 1;            /* Indicador de ritmo */
    unsigned char    bbi : 1;         /* Indicador de corchete inicial */
    unsigned char    ebi : 1;         /* Indicador de corchete final */
    unsigned char    cdi : 1;         /* Indicador de cambio dirección */
    unsigned char    : 1;            /* Campo reservado */
    unsigned char    csi : 1;         /* Indicador de selección código */
    unsigned char    edi : 1;         /* Campo reservado */
    unsigned char    pdi : 1;         /* Indicador de datos cifrados */
    unsigned char    : 1;            /* Indicador de datos rellenos */
    } lua_rh;                          /* Campo reservado */

struct LUA_FLAG1 {
    unsigned char    bid_enable : 1;  /* LUA_FLAG1 */
    unsigned char    reserv1 : 1;     /* Indicador anuncio activado */
    unsigned char    close_abend : 1; /* reservado */
    unsigned char    nowait : 1;      /* Indicador de cierre inmediato */
    unsigned char    sscp_exp : 1;    /* Indicador de no esperar datos */
    unsigned char    sscp_norm : 1;   /* Flujo rápido de SSCP */
    unsigned char    lu_exp : 1;      /* Flujo normal de SSCP */
    unsigned char    lu_norm : 1;     /* Flujo rápido de LU */
    } lua_flag1;                       /* Flujo normal de lu */

unsigned char    lua_message_type;   /* Tipo mandato mensaje SNA */

struct LUA_FLAG2 {
    unsigned char    bid_enable : 1;  /* LUA_FLAG2 */
    unsigned char    async : 1;       /* Indicador anuncio activado */
    unsigned char    : 2;             /* Indicador de finalización */
    unsigned char    sscp_exp : 1;    /* asíncrona de verbo */
    unsigned char    sscp_norm : 1;   /* reservado */
    unsigned char    lu_exp : 1;      /* Flujo rápido de SSCP */
    unsigned char    lu_norm : 1;     /* Flujo normal de SSCP */
    } lua_flag2;                       /* Flujo rápido de LU */
    unsigned char    lua_resv56[7];    /* Flujo normal de lu */
    unsigned char    lua_encr_decr_option; /* Campo reservado */
} ;                                     /* Opción de criptografía */

```



Formato de VCB (bloque de control de verbos) LUA

En la siguiente lista se explican los campos de estas estructuras de datos.

lua_verb

Lo identifica como un verbo LUA.

Los valores posibles son:

LUA_VERB_RUI

Verbo RUI.

LUA_VERB_SLI

Verbo SLI.

lua_verb_length

Longitud del bloque de control de verbos (VCB).

lua_prim_rc

Código de retorno primario definido por LUA.

lua_sec_rc

Código de retorno secundario definido por LUA.

lua_opcode

Código de operación de verbo que identifica el verbo LUA que se emite.

lua_correlator

Correlacionador de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza este parámetro.

lua_luname

Nombre de LU utilizado por la sesión LUA (en ASCII). Puede ser un nombre de LU o un nombre de agrupación de LU; para obtener más información, consulte "RUI_INIT" en la página 66 o "SLI_OPEN" en la página 121.

AIX, LINUX

Para RUI_INIT_PRIMARY, debe coincidir con el parámetro *dslu_name* de una LU en sentido descendente configurada para utilizarse con la pasarela SNA (o una LU en sentido descendente creada implícitamente definiendo una plantilla de LU en sentido descendente).

lua_extension_list_offset

Este campo está reservado.

lua_cobol_offset

Este campo está reservado.

lua_sid Identificador de sesión de la sesión LUA en la que se emite este verbo.

lua_max_length

La longitud del almacenamiento intermedio que se proporciona a RUI_READ, RUI_INIT_PRIMARY o SLI_RECEIVE para recibir datos o la longitud total de una RU en espera devuelta a RUI_BID.

lua_data_length

Longitud de los datos que se van a enviar, o longitud real de los datos recibidos.

Formato de VCB (bloque de control de verbos) LUA

lua_data_ptr

Puntero a los datos que se van a enviar o almacenamiento intermedio de datos para la recepción de datos.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI o SLI, establezca este campo como un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI o WinSLI, este campo está reservado.

████████

lua_th Estructura de datos que contiene la cabecera de transmisión (TH) del mensaje enviado o recibido, como se muestra a continuación:

lua_th.flags_fid

Identificación de formato tipo 2: 4 bits

lua_th.flags_mpf

Campo de correlación de segmentación: 2 bits

lua_th.flags_odai

Indicador de asignador OAF-DAF (campo de dirección de origen - campo de dirección de destino)

lua_th.flags_efi

Indicador de flujo rápido

lua_th.daf

DAF (campo de dirección de destino)

lua_th.oaf

OAF (campo de dirección de origen)

lua_th.snf

Campo de número de secuencia

lua_rh Estructura de datos que contiene la cabecera de petición/respuesta (RH) del mensaje enviado o recibido, como se indica a continuación:

lua_rh.rrl

Indicador de petición/respuesta

lua_rh.ruc

Categoría de RU: 2 bits

lua_rh.fi

Indicador de formato

lua_rh.sdi

Indicador de datos de detección incluidos

lua_rh.bci

Indicador de principio de cadena

Formato de VCB (bloque de control de verbos) LUA

| | |
|------------------------------|---|
| <i>lua_rh.eci</i> | Indicador de fin de cadena |
| <i>lua_rh.dr1i</i> | Indicador de respuesta concreta 1 |
| <i>lua_rh.dr2i</i> | Indicador de respuesta concreta 2 |
| <i>lua_rh.ri</i> | Indicador de respuesta de excepción (para una petición) o indicador de tipo de respuesta (para una respuesta) |
| <i>lua_rh.qri</i> | Indicador de respuesta en cola |
| <i>lua_rh.pi</i> | Indicador de ritmo |
| <i>lua_rh.bbi</i> | Indicador de inicio de corchete |
| <i>lua_rh.ebi</i> | Indicador de fin de corchete |
| <i>lua_rh.cdi</i> | Indicador de cambio de dirección activado |
| <i>lua_rh.csi</i> | Indicador de selección de código |
| <i>lua_rh.edi</i> | Indicador de datos cifrados |
| <i>lua_rh.pdi</i> | Indicador de datos rellenados |
| <i>lua_flag1</i> | Estructura de datos que contiene indicadores para mensajes suministrados por la aplicación: |
| <i>lua_flag1.bid_enable</i> | Indicador de anuncio activado |
| <i>lua_flag1.close_abend</i> | Indicador de cierre inmediato |
| <i>lua_flag1.nowait</i> | Indicador de no esperar datos |
| <i>lua_flag1.sscp_exp</i> | Flujo rápido de SSCP |
| <i>lua_flag1.sscp_norm</i> | Flujo normal de SSCP |
| <i>lua_flag1.lu_exp</i> | Flujo rápido de LU |
| <i>lua_flag1.lu_norm</i> | Flujo normal de LU |
| <i>lua_message_type</i> | Tipo de mensaje SNA recibido por un verbo RUI_READ o SLI_RECEIVE (o indicado a un verbo RUI_BID o SLI_BID) |

Formato de VCB (bloque de control de verbos) LUA

lua_flag2

Estructura de datos que contiene indicadores para los mensajes devueltos por LUA, como se muestra a continuación:

lua_flag2.bid_enable

Indicador de anuncio activado

lua_flag2.async

Indicador de finalización asíncrona de verbo

lua_flag2.sscp_exp

Flujo rápido de SSCP

lua_flag2.sscp_norm

Flujo normal de SSCP

lua_flag2.lu_exp

Flujo rápido de LU

lua_flag2.lu_norm

Flujo normal de LU

lua_encr_decr_option

Opción de criptografía. Para SLI, este parámetro está reservado y debe establecerse en cero.

Estructura de datos específica

La estructura de datos *especifica* se incluye para los verbos siguientes:

- RUI_BID
- Formato ampliado de RUI_INIT
- SLI_BID
- SLI_OPEN
- SLI_SEND

```
union LUA_SPECIFIC
{
struct SLI_OPEN  open;
unsigned char   lua_sequence_number[2];
unsigned char   lua_peek_data[12];
struct RUI_INIT  init;
} ;
```

AIX, LINUX

```
struct SLI_OPEN
{
    unsigned char  lua_init_type;           /* Tipo de iniciación de sesión*/
    unsigned char  lua_session_type;       /* Cómo procesar UNBIND s.pra1.*/
    AP_UINT16      lua_wait;               /* Tiempo esp. reintento secun.*/

    struct LUA_EXT_ENTRY
    {
        unsigned char  lua_routine_type;   /* Tiempo de rutina ampliación */
        unsigned long  lua_routine_ptr;    /* Puntero a rutina ampliación */
    } lua_open_extension[MAX_EXTENSIONS];

    char reserved[93];                     /* Relleno */
    unsigned char  lua_ending_delim;       /* Delimitador lista ampliación*/
};
```

WINDOWS

```

struct SLI_OPEN
{
    unsigned char  lua_init_type;           /* Tipo de iniciación de sesión*/
    unsigned char  lua_session_type;       /* Cómo procesar UNBIND s.pral.*/
    AP_UINT16      lua_wait;               /* Tiempo esp. reintento secun.*/

    struct LUA_EXT_ENTRY
    {
        unsigned char lua_routine_type;    /* Tiempo de rutina ampliación */
        unsigned char lua_module_name[9];  /* Nombre de módulo DLL ampli. */
        unsigned char lua_procedure_name[33]; /* Nombre de proced. a llamar */
    } lua_open_extension[MAX_EXTENSIONS];

    char reserved[93];                     /* Relleno */
    unsigned char lua_ending_delim;        /* Delimitador lista ampliación*/
};

```

```

struct RUI_INIT
{
    unsigned char  rui_init_format;
    unsigned char  lua_puname[8];
    unsigned char  lua_lunumber;
    unsigned char  wait_for_link;
};

```

Para RUI_BID y SLI_BID, esta estructura de datos contiene el campo siguiente:

lua_peek_data

Hasta 12 bytes de los datos que están en espera de ser leídos.

AIX, LINUX

Para el formato ampliado del verbo RUI_INIT, esta estructura de datos contiene los siguientes campos. Para ver más información sobre el formato ampliado de RUI_INIT, consulte "RUI_INIT" en la página 66.

rui_init_format

Reservado: este parámetro debe tener el valor 0 (cero).

lua_puname

Nombre de la PU local propietaria de la LU que debe utilizarse para esta sesión. El nombre de PU se debe especificar en la definición de una LS o de una PU interna en la configuración de Communications Server para Linux.

lua_lunumber

Número de la LU que debe utilizarse para esta sesión. Debe coincidir con el número de LU de tipo 0-3 LU configurada para el nombre de PU especificado por *lua_puname*.

wait_for_link

Normalmente, si la aplicación emite RUI_INIT para una LU que no se puede utilizar actualmente porque el enlace de las comunicaciones subyacente está inactivo, el verbo RUI_INIT falla. Establezca este parámetro en 1 para alterar temporalmente este funcionamiento por

Formato de VCB (bloque de control de verbos) LUA

omisión de modo que LUA espere a que el enlace y la LU estén activos antes de finalizar RUI_INIT o en 0 (cero) para utilizar el funcionamiento por omisión.

WINDOWS

El formato ampliado del verbo RUI_INIT no se aplica a Windows. La estructura de datos RUI_INIT no se utiliza.



Para SLI_OPEN, esta estructura de datos contiene los campos siguientes. Consulte "SLI_OPEN" en la página 121 para obtener información detallada sobre estos parámetros.

lua_init_type

Especifica cómo LUA inicia la sesión (si la primaria o la secundaria es la responsable del inicio de sesión y la secuencia de mensajes SNA necesaria).

lua_session_type

Especifica cómo LUA debe procesar un UNBIND de tipo X'01' (normal): si se trata de una sesión normal o dedicada.

lua_wait

Tiempo de espera de reintento (en segundos) para el arranque de sesión de inicio secundario.

lua_open_extension

Estructura que contiene información sobre las rutinas de ampliación de SLI_OPEN de la aplicación, si existe alguna.

lua_open_extension.lua_routine_type

Tipo de rutina de ampliación (BIND, SDT o STSN).

AIX, LINUX

lua_open_extension.lua_routine_ptr

Puntero hacia el punto de entrada de la rutina de ampliación.

WINDOWS

lua_open_extension.lua_module_name

Nombre de la DLL que contiene el módulo de ampliación.

lua_open_extension.lua_procedure_name

Nombre de procedimiento a llamar dentro de la DLL del módulo de ampliación.



lua_ending_delim

La interfaz SLI de Communications Server para Linux no utiliza este parámetro; se proporciona por compatibilidad con aplicaciones escritas originalmente para otras implementaciones de SLI.

Formato de VCB (bloque de control de verbos) LUA

Para SLI_SEND, esta estructura de datos contiene el campo siguiente.

lua_sequence_number

Número de secuencia de la RU que LUA utiliza para enviar los datos (o de la primera RU, si los datos necesitan una cadena de RU). Se guarda en formato de línea.

Formato de VCB (bloque de control de verbos) LUA

Capítulo 4. Verbos RUI

Este capítulo contiene una descripción de cada verbo RUI LUA. Se proporciona la información siguiente para cada verbo:

- Propósito del verbo.
- Parámetros (campos del VCB) suministrados a LUA y devueltos por LUA. La descripción de cada parámetro incluye información sobre los valores válidos para el parámetro y toda la información adicional que sea necesaria.
- Interacción con otros verbos.
- Información adicional sobre el uso del verbo.

Para obtener detalles del bloque de control de verbos (VCB) que se utiliza para todos los verbos, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Las constantes simbólicas se definen en los archivos de cabecera `lua_c.h` y `values_c.h` (sistema operativo AIX o Linux) o `winlua.h` (sistema operativo Windows) para muchos valores de parámetros. Por motivos de portabilidad, utilice la constante simbólica y no el valor numérico cuando establezca valores para los parámetros proporcionados o cuando pruebe los valores de los parámetros devueltos. El archivo `values_c.h` también incluye las definiciones de tipos de parámetro, como `AP_UINT16`, que se utilizan en los VCB LUA.

Los parámetros marcados como “reservados” siempre deben tener el valor 0 (cero).

RUI_BID

La aplicación utiliza el verbo `RUI_BID` para determinar cuándo un mensaje recibido está en espera de ser leído. Esto permite que la aplicación determine qué datos, si los hay, están disponibles antes de emitir el verbo `RUI_READ`.

Cuando hay un mensaje disponible, el verbo `RUI_BID` vuelve con detalles del flujo de mensajes en el que se ha recibido, el tipo de mensaje, la TH y RH del mensaje y hasta 12 bytes de datos del mensaje.

La diferencia principal entre `RUI_BID` y `RUI_READ` es que `RUI_BID` permite que la aplicación consulte los datos sin eliminarlos de la cola de mensajes de entrada, de modo que se puede acceder a dichos datos más adelante. El verbo `RUI_READ` elimina el mensaje de la cola, por lo que, una vez que la aplicación ha leído los datos, debe procesarlos.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

LUA_OPCODE_RUI_BID

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Éste debe coincidir con el nombre de LU de una sesión LUA activa (devuelto en el verbo RUI_INIT o RUI_INIT_PRIMARY).

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Éste debe coincidir con un ID de sesión devuelto en un verbo RUI_INIT o RUI_INIT_PRIMARY anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_max_length

Número total de bytes del mensaje en espera.

lua_data_length

Número de bytes de los datos devueltos en el parámetro *lua_peek_data*; de 0 a 12.

lua_th TH del mensaje recibido.

lua_rh RH del mensaje recibido.

lua_message_type

Tipo de mensaje recibido; puede ser uno de los siguientes:

LUA_MESSAGE_TYPE_LU_DATA
 LUA_MESSAGE_TYPE_SSCP_DATA
 LUA_MESSAGE_TYPE_RSP
 LUA_MESSAGE_TYPE_BID
 LUA_MESSAGE_TYPE_BIND
 LUA_MESSAGE_TYPE_BIS
 LUA_MESSAGE_TYPE_CANCEL
 LUA_MESSAGE_TYPE_CHASE
 LUA_MESSAGE_TYPE_CLEAR
 LUA_MESSAGE_TYPE_CRV
 LUA_MESSAGE_TYPE_LUSTAT_LU
 LUA_MESSAGE_TYPE_LUSTAT_SSCP
 LUA_MESSAGE_TYPE_QC
 LUA_MESSAGE_TYPE_QEC
 LUA_MESSAGE_TYPE_RELQ
 LUA_MESSAGE_TYPE_RTR
 LUA_MESSAGE_TYPE_SBI
 LUA_MESSAGE_TYPE_SHUTD
 LUA_MESSAGE_TYPE_SIGNAL
 LUA_MESSAGE_TYPE_SDT
 LUA_MESSAGE_TYPE_STSN
 LUA_MESSAGE_TYPE_UNBIND

AIX, LINUX

Los valores siguientes sólo se pueden devolver a una aplicación RUI primaria (la que ha iniciado la sesión utilizando RUI_INIT_PRIMARY):

LUA_MESSAGE_TYPE_INIT_SELF

LUA_MESSAGE_TYPE_NOTIFY

LUA_MESSAGE_TYPE_TERM_SELF

*lua_flag2*

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han recibido los datos:

*lua_flag2.sscp_exp**lua_flag2.lu_exp**lua_flag2.sscp_norm**lua_flag2.lu_norm**lua_peek_data*

Los 12 primeros bytes de los datos del mensaje (o todos los datos del mensaje si son menos de 12 bytes)

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo RUI_TERM mientras este verbo estaba pendiente.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_BID_ALREADY_ENABLED

Se ha rechazado el verbo RUI_BID porque todavía había un verbo RUI_BID anterior pendiente para esta sesión. No puede haber más de un verbo RUI_BID pendiente para una misma sesión.

| |
|------------|
| AIX, LINUX |
|------------|

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

**LUA_RESERVED_FIELD_NOT_ZERO**

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

LUA_NO_RUI_SESSION

Un verbo RUI_INIT o RUI_INIT_PRIMARY aún no se ha completado satisfactoriamente para el nombre de LU especificado en este verbo o ha fallado la sesión.

Respuesta negativa enviada al sistema principal: El código de retorno siguiente indica que Communications Server para Linux ha detectado un error en los datos recibidos del sistema principal. En lugar de pasar el mensaje recibido a la aplicación en un verbo RUI_READ, Communications Server para Linux elimina el mensaje (y el resto de la cadena si está en una cadena) y envía una respuesta negativa al sistema principal. LUA informa a la aplicación en un verbo RUI_READ o RUI_BID posterior de que se ha enviado una respuesta negativa.

lua_prim_rc
LUA_NEGATIVE_RSP

lua_sec_rc

El código de retorno secundario contiene el código de detección enviado al sistema principal en la respuesta negativa. Consulte "Información de SNA" en la página 36, para ver información sobre cómo interpretar los valores de código de detección que pueden devolverse.

Un código de retorno secundario de 0 (cero) indica que ahora, después de un RUI_WRITE anterior de una respuesta negativa a un mensaje en mitad de una cadena, Communications Server para Linux ha recibido y eliminado todos los mensajes de esta cadena.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

LUA_INVALID_PROCESS

El proceso de sistema operativo que ha emitido este verbo no era el mismo proceso que ha emitido el verbo RUI_INIT o

RUI_INIT_PRIMARY para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si la sesión se ha iniciado utilizando RUI_INIT (no RUI_INIT_PRIMARY) y el código de retorno secundario no es LUA_RUI_LOGIC_ERROR, se puede reinicializar esta LU utilizando RUI_REINIT. Si no se reinicializa, se debe emitir RUI_TERM para que se pueda emitir RUI_INIT o RUI_INIT_PRIMARY para la misma LU.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

El verbo RUI_INIT o RUI_INIT_PRIMARY se debe completar satisfactoriamente para que se pueda emitir este verbo.

No puede haber más de un verbo RUI_BID pendiente simultáneamente para cada sesión.

Una vez que el verbo RUI_BID ha finalizado correctamente, puede volver a emitirse enviando el parámetro *lua_flag1.bid_enable* en otro verbo RUI_READ. Si el verbo se volverá a emitir de este modo, el programa de aplicación no debe liberar ni modificar el almacenamiento asociado con el registro de verbo RUI_BID.

Si llega un mensaje del sistema principal cuando hay un verbo RUI_READ y un verbo RUI_BID pendientes, RUI_READ finaliza y RUI_BID se deja en proceso.

Uso y restricciones

Cada mensaje que llega se anuncia una sola vez. Cuando el verbo RUI_BID ha indicado que hay datos en espera en un determinado flujo de sesión, la aplicación debe emitir el verbo RUI_READ para recibir los datos. Ningún verbo RUI_BID posterior informará de la llegada de datos en ese flujo de sesión hasta que el mensaje que se ha anunciado se haya aceptado emitiendo un verbo RUI_READ.

Los siguientes elementos describen la diferencia entre los parámetros *lua_max_length* y *lua_data_length* devueltos en este verbo:

- El parámetro *lua_max_length* indica la longitud del mensaje en espera. Cuando se emite el verbo RUI_READ para aceptar el mensaje, la aplicación debe suministrar un almacenamiento intermedio de datos de como mínimo este tamaño a fin de garantizar que el mensaje puede recibirse sin que se trunque.
- El parámetro *lua_data_length* indica la longitud de los datos de *lua_peek_data*. Si es inferior a 12 (lo cual indica que el mensaje en espera tiene menos de 12 bytes), los demás bytes de *lua_peek_data* no están definidos y la aplicación no debe intentar examinarlos.

RUI_INIT

El verbo RUI_INIT establece la sesión de SSCP-LU para una LU determinada o establece una sesión de SSCP-LU para la LU utilizada menos recientemente de una agrupación de LU determinada.

AIX, LINUX

En general, la aplicación especifica el nombre de una LU o una agrupación de LU que se debe utilizar para la sesión. Communications Server para Linux también proporciona un formato ampliado de RUI_INIT, en el que la aplicación puede identificar la LU especificando el nombre de PU y el número de LU en lugar del nombre de LU; otras implementaciones de LUA no soportan esta función. Las diferencias entre las versiones normal y ampliada de RUI_INIT se indican, cuando procede, en las descripciones de los parámetros en este mismo apartado.

La aplicación RUI debe utilizar RUI_INIT_PRIMARY en lugar de RUI_INIT si actúa como SNA primaria para las comunicaciones con una LU en sentido descendente.



Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

AIX, LINUX

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

Para garantizar la compatibilidad con otras implementaciones de LUA, el valor `sizeof(LUA_COMMON)` también se acepta si utiliza el formato estándar de RUI_INIT en lugar del formato ampliado.

WINDOWS

Establézcalo en `sizeof(LUA_COMMON)`.



lua_opcode

LUA_OPCODE_RUI_INIT

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU o agrupación de LU para la que desea iniciar la sesión. Debe coincidir con el nombre de una LU de tipo 0–3, o de una agrupación de LU, configurada para Communications Server para Linux. El nombre se utiliza de la siguiente manera:

- Si el nombre es el nombre de una LU que no está en una agrupación, Communications Server para Linux intenta iniciar la sesión utilizando esta LU. Una aplicación puede iniciar varias sesiones utilizando varios verbos RUI_INIT con una LU diferente para cada verbo; no puede iniciar más de una sesión para la misma LU.
- Si el nombre es el nombre de una agrupación de LU, o el nombre de una LU de una agrupación, Communications Server para Linux intenta iniciar la sesión utilizando la LU mencionada si está disponible o, de lo contrario, la LU utilizada menos recientemente de la agrupación. Una aplicación puede iniciar varias sesiones utilizando la misma agrupación; Communications Server para Linux asignará una LU diferente de la agrupación para cada sesión. El nombre de la LU real utilizada para la sesión es un parámetro devuelto en el verbo RUI_INIT.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

| |
|------------|
| AIX, LINUX |
|------------|

La aplicación puede utilizar el formato ampliado de RUI_INIT para identificar la LU por nombre de PU y número de LU, en lugar de por su nombre de LU. Para ello, establezca *lua_luname* con ocho ceros binarios y especifique el nombre de PU y el número de LU en los parámetros *lua_puname* y *lua_lunumber*.

| |
|--|
| |
|--|

lua_post_handle

| |
|------------|
| AIX, LINUX |
|------------|

Puntero a una rutina de devolución de llamada. Si el verbo se completa de forma asíncrona, LUA llamará esta rutina para indicar la finalización del verbo.

| |
|---------|
| WINDOWS |
|---------|

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.

| |
|--|
| |
|--|

Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

lua_encr_decr_option

Opción de criptografía a nivel de sesión. Communications Server para Linux acepta los dos valores siguientes:

0 No se utiliza criptografía a nivel de sesión.

128 El programa de aplicación realiza el cifrado y descifrado.

Cualquier otro valor generará el código de retorno LUA_ENCR_DECR_LOAD_ERROR. (La implementación de LUA de OS/2 Extended Editions soporta los valores en el rango 1 a 127, que indican rutinas de cifrado y descifrado definidas por el usuario, pero Communications Server para Linux no los soporta.)

| |
|------------|
| AIX, LINUX |
|------------|

Los parámetros siguientes sólo se utilizan si el parámetro *lua_luname* está definido con ocho ceros binarios (el formato ampliado de RUI_INIT). Si *lua_luname* especifica el nombre de LU (el formato estándar de RUI_INIT), estos parámetros están reservados.

lua_puname

Nombre de la PU propietaria de la LU que debe utilizarse para esta sesión. El nombre debe estar en ASCII, rellenado con espacios por la derecha (0x20). Debe coincidir con un nombre de PU definido en la configuración de Communications Server para Linux.

lua_lunumber

Número de la LU que debe utilizarse para esta sesión. Debe coincidir con el número de LU de tipo 0-3 LU configurado para utilizar la PU especificada.

Una aplicación puede iniciar varias sesiones utilizando varios verbos RUI_INIT con una LU diferente para cada verbo; no puede iniciar más de una sesión para la misma LU.

wait_for_link

Normalmente, si la aplicación emite RUI_INIT para una LU que no se puede utilizar actualmente porque el enlace de las comunicaciones subyacente está inactivo, el verbo RUI_INIT falla. Establezca este parámetro en 1 para alterar temporalmente este funcionamiento por omisión de modo que LUA espere a que el enlace y la LU estén activos antes de finalizar RUI_INIT o en 0 (cero) para utilizar el funcionamiento por omisión.



Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 si ha finalizado de manera síncrona. (RUI_INIT siempre finaliza de manera asíncrona, a menos que devuelva un error como, por ejemplo, LUA_PARAMETER_CHECK.)

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, LUA devuelve los parámetros siguientes.

lua_prim_rc
LUA_OK

lua_sid Identificador de sesión para la nueva sesión. Verbos posteriores pueden utilizarlo para identificar esta sesión.

lua_luname
Nombre de la LU utilizada por la nueva sesión. Si el nombre de LU en los parámetros de petición especificaba una agrupación de LU o si la aplicación utilizaba el formato ampliado de RUI_INIT y especificaba el nombre de PU y el número de LU en lugar del nombre de LU, Communications Server para Linux utiliza este parámetro para devolver el nombre de la LU real asignada a la sesión. Los verbos posteriores deben utilizar este nombre devuelto (no el nombre especificado en los parámetros de petición) para identificar la sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
LUA_TERMINATED
Se ha emitido un verbo RUI_TERM antes de que finalizara RUI_INIT.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
Los valores posibles son:

LUA_INVALID_LUNAME
No se ha encontrado la LU identificada por el parámetro *lua_luname* en ningún nodo activo. Compruebe que el nombre de LU o el nombre de agrupación de LU está definido en el archivo de configuración y que el nodo en el que está configurado se ha iniciado.

AIX, LINUX

LUA_INVALID_POST_HANDLE
El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

| |
|------------|
| AIX, LINUX |
|------------|

Los parámetros siguientes sólo se utilizan si el parámetro *lua_luname* está definido con ocho ceros binarios (el formato ampliado de RUI_INIT). Si *lua_luname* especifica el nombre de LU (el formato estándar de RUI_INIT), estos parámetros están reservados.

LUA_INVALID_FORMAT

El parámetro reservado *ru_i_init_format* tiene un valor distinto de cero.

LUA_INVALID_PUNAME

El parámetro *lua_puname* no coincidía con ningún nombre de PU definido en la configuración de Communications Server para Linux.

LUA_INVALID_LUNUMBER

El parámetro *lua_lunumber* no coincide con el número de una LU de tipo 0–3 definida para utilizar la PU especificada.



Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

LUA_DUPLICATE_RUI_INIT

En este momento se está procesando un verbo RUI_INIT para esta sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_COMMAND_COUNT_ERROR

El verbo especifica el nombre de una agrupación de LU o el nombre de una LU de una agrupación, pero ya se están utilizando todas las LU de la agrupación.

LUA_ENCR_DECR_LOAD_ERROR

El verbo especifica un valor para *lua_encr_decr_option* distinto de 0 ó 128.

LUA_INVALID_PROCESS

Otro proceso está utilizando la LU especificada por el parámetro *lua_luname*.

LUA_LINK_NOT_STARTED

La conexión con el sistema principal no se ha iniciado; ninguno de los enlaces que puede utilizar está activo.

(cualquier otro valor)

Cualquier otro código de retorno secundario que aparezca aquí es un código de detección SNA. Para ver información sobre cómo interpretar los códigos de detección SNA que pueden devolverse, consulte "Información de SNA" en la página 36.

Los siguientes valores de código de detección son específicos de Communications Server para Linux y pueden indicar discrepancias entre la configuración de Communications Server para Linux y la configuración de sistema principal:

0x10020000

El sistema principal no ha enviado un mensaje de activación de unidad física (ACTPU) para la PU propietaria de la LU solicitada.

0x10110000

El sistema principal no ha enviado un mensaje ACTLU para la LU solicitada. Normalmente esto indica que la LU no está configurada en el sistema principal.

0x10120000

El sistema principal no ha enviado un mensaje ACTLU para la LU solicitada. El sistema principal soporta DDDL (definición dinámica de LU dependientes), pero el proceso DDDL para esta LU ha fallado.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

Este código de retorno indica una de las siguientes condiciones:

- El software Remote API Client no se ha iniciado. Inicie Remote API Client antes de ejecutar la aplicación.
- No hay nodos de Communications Server para Linux activos. Para poder utilizar los verbos LUA, debe haberse iniciado el nodo local propietario de la LU solicitada o un nodo local propietario de una o varias LU de la agrupación de LU solicitada. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si el código de retorno secundario no es `LUA_RUI_LOGIC_ERROR`, esta LU puede reiniciarse utilizando un verbo `RUI_REINIT`. Si no se reinicializa, debe emitirse un verbo `RUI_TERM` para que se pueda emitir un verbo `RUI_INIT` para la misma LU.

lua_sec_rc

LUA_LU_COMPONENT_DISCONNECTED

La sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

██████████

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

Este verbo debe ser el primer verbo LUA emitido para la sesión.

Hasta que el verbo haya finalizado correctamente, el único otro verbo LUA que puede emitirse para esta sesión es `RUI_TERM` (que finalizará un verbo `RUI_INIT` pendiente).

Todos los demás verbos emitidos en esta sesión deben identificar la sesión utilizando uno de los parámetros devueltos de este verbo:

- El identificador de sesión, devuelto a la aplicación en el parámetro *lua_sid*
- El nombre de LU, devuelto a la aplicación en el parámetro *lua_luname*

Uso y restricciones

El verbo `RUI_INIT` finaliza cuando se recibe un mensaje `ACTLU` del sistema principal. Si es necesario, el verbo espera de manera indefinida. Si ya se ha recibido un mensaje `ACTLU` antes del verbo `RUI_INIT`, LUA envía un mensaje

NOTIFY al sistema principal para informar de que ya se puede utilizar la LU. Ni ACTLU ni NOTIFY son visibles para la aplicación LUA.

Una vez que el verbo RUI_INIT ha finalizado correctamente, esta sesión utiliza la LU para la que se ha iniciado la sesión. Ninguna otra sesión LUA (de esta o de otra aplicación) puede utilizar la LU hasta que se ha emitido el verbo RUI_TERM.

Si el verbo RUI_INIT vuelve con un código de retorno primario LUA_IN_PROGRESS, el identificador de sesión se devolverá en el parámetro *lua_sid*. Este identificador de sesión es el mismo que el devuelto cuando el verbo finaliza de forma correcta y puede utilizarse con el verbo RUI_TERM para finalizar un verbo RUI_INIT pendiente.

RUI_INIT_PRIMARY

AIX, LINUX

El verbo RUI_INIT_PRIMARY establece la sesión de SSCP-LU para una aplicación SNA primaria que se comunica con una LU en sentido descendente. (Si la aplicación RUI actúa como SNA secundaria y se comunica con una LU de sistema principal, debe utilizar RUI_INIT en lugar de RUI_INIT_PRIMARY.)

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Establézcalo en `sizeof(LUA_COMMON)`.

lua_opcode

LUA_OPCODE_RUI_INIT_PRIMARY

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU para la que desea iniciar la sesión. Éste debe coincidir con el nombre de una LU en sentido descendente configurada para utilizarse con la pasarela SNA o una LU creada implícitamente desde una plantilla de LU en sentido descendente.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_max_length

Longitud de un almacenamiento intermedio proporcionado para recibir una copia de la RU ACTLU(+RSP) recibida de la PU en sentido descendente. Si la aplicación no necesita recibir esta información, puede especificar un puntero nulo en el parámetro *lua_data_ptr*, en cuyo caso no necesita proporcionar un almacenamiento intermedio de datos.

lua_data_ptr

Puntero al almacenamiento intermedio proporcionado para recibir una copia de la RU ACTLU(+RSP) recibida de la PU en sentido descendente. Si la aplicación no necesita recibir esta información, puede especificar un puntero nulo y no se devolverá la información.

lua_post_handle

Puntero a una rutina de devolución de llamada. Si el verbo se completa de forma asíncrona, LUA llamará a esta rutina para indicar la finalización del verbo. Para ver más información, consulte el Capítulo 2, "Diseño y desarrollo de aplicaciones LUA", en la página 13.

lua_encr_decr_option

Opción de criptografía a nivel de sesión. Communications Server para Linux acepta los dos valores siguientes:

0 No se utiliza criptografía a nivel de sesión.

128 El programa de aplicación realiza el cifrado y descifrado.

Cualquier otro valor generará el código de retorno LUA_ENCR_DECR_LOAD_ERROR. (La implementación de LUA de OS/2 Extended Editions soporta los valores en el rango 1 a 127, que indican rutinas de cifrado y descifrado definidas por el usuario, pero Communications Server para Linux no los soporta.)

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 si ha finalizado de manera síncrona. (RUI_INIT_PRIMARY se completará siempre de forma asíncrona, a menos que devuelva un error, por ejemplo LUA_PARAMETER_CHECK.)

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, LUA devuelve los parámetros siguientes.

lua_prim_rc

LUA_OK

lua_sid Identificador de sesión para la nueva sesión. Verbos posteriores pueden utilizarlo para identificar esta sesión.

lua_data_length

Longitud de la RU ACTLU(+RSP) recibida de la PU en sentido descendente. LUA coloca los datos en el almacenamiento intermedio especificado por *lua_data_ptr*.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo RUI_TERM antes de que RUI_INIT_PRIMARY se haya completado.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_INVALID_LUNAME

No se ha podido encontrar en ningún nodo activo la LU identificada por el parámetro *lua_luname*. Compruebe que el nombre de LU se haya definido en el archivo de configuración y que se haya iniciado el nodo en el que se ha configurado.

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

LUA_DUPLICATE_RUI_INIT_PRIMARY

Actualmente se está procesando un verbo RUI_INIT_PRIMARY para esta sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_ENCR_DECR_LOAD_ERROR

El verbo especifica un valor para *lua_encr_decr_option* distinto de 0 ó 128.

LUA_INVALID_PROCESS

Otro proceso está utilizando la LU especificada por el parámetro *lua_luname*.

RUI_INIT_PRIMARY

(cualquier otro valor)

Cualquier otro código de retorno secundario que aparezca aquí es un código de detección SNA. Para ver información sobre cómo interpretar los códigos de detección SNA que pueden devolverse, consulte “Información de SNA” en la página 36.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

Este código de retorno indica una de las siguientes condiciones:

- El software Remote API Client no se ha iniciado. Inicie Remote API Client antes de ejecutar la aplicación.
- No hay nodos de Communications Server para Linux activos. Para poder utilizar los verbos LUA, se debe iniciar el nodo local propietario de la LU en sentido descendente solicitada. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Se debe emitir RUI_TERM para que se pueda emitir otro RUI_INIT_PRIMARY para la misma LU.

lua_sec_rc

LUA_LU_COMPONENT_DISCONNECTED

La sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

Este verbo debe ser el primer verbo LUA emitido para la sesión.

Hasta que este verbo se haya completado satisfactoriamente, el único otro verbo LUA que se puede emitir para esta sesión es RUI_TERM (que terminará un RUI_INIT_PRIMARY pendiente).

Todos los demás verbos emitidos en esta sesión deben identificar la sesión utilizando uno de los parámetros siguientes de este verbo:

- El identificador de sesión, devuelto a la aplicación en el parámetro *lua_sid*
- El nombre de LU, proporcionado por la aplicación en el parámetro *lua_luname*

Uso y restricciones

El verbo RUI_INIT_PRIMARY se completa después de que se haya recibido una respuesta ACTLU afirmativa de la LU en sentido descendente. Si es necesario, el verbo espera de manera indefinida. Si la aplicación necesita comprobar el contenido de esta respuesta ACTLU afirmativa, puede hacerlo proporcionando un almacenamiento intermedio de datos en RUI_INIT_PRIMARY (utilizando los parámetros *lua_max_length* y *lua_data_ptr*) en el que Communications Server para Linux devuelve el contenido del mensaje recibido.

Una vez que el verbo RUI_INIT_PRIMARY se ha completado satisfactoriamente, esta sesión utiliza la LU para la que se ha iniciado la sesión. Ninguna otra sesión LUA (de esta u otra aplicación) puede utilizar la LU hasta que se emite el verbo RUI_TERM o hasta que se recibe el código de retorno primario LUA_SESSION_FAILURE.

Si el verbo RUI_INIT_PRIMARY devuelve un código de retorno primario LUA_IN_PROGRESS, se devolverá el ID de sesión en el parámetro *lua_sid*. Este ID de sesión es el mismo que el devuelto cuando el verbo se completa satisfactoriamente y se puede utilizar con el verbo RUI_TERM para terminar un verbo RUI_INIT_PRIMARY pendiente.



RUI_PURGE

El verbo RUI_PURGE cancela un RUI_READ anterior. Un RUI_READ puede esperar indefinidamente si se envía sin utilizar la opción *lua_flag1.nowait* (retorno inmediato) y no hay datos disponibles en el flujo especificado; RUI_PURGE fuerza el retorno del verbo en espera (con el código de retorno primario LUA_CANCELLED).

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Establézcalo en `sizeof(LUA_COMMON)`.

lua_opcode

LUA_OPCODE_RUI_PURGE

RUI_PURGE

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Éste debe coincidir con el nombre de LU de una sesión LUA activa, tal como se devuelve en el verbo RUI_INIT o RUI_INIT_PRIMARY.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Éste debe coincidir con un ID de sesión devuelto en un verbo RUI_INIT o RUI_INIT_PRIMARY anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_data_ptr

Puntero al VCB RUI_READ que se va a eliminar.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.

■■■■■

Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, se devuelven los parámetros siguientes:

lua_prim_rc
LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo RUI_TERM mientras este verbo estaba pendiente.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* tiene el valor 0 (cero).

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

RUI_PURGE

lua_sec_rc

LUA_NO_RUI_SESSION

Un verbo RUI_INIT o RUI_INIT_PRIMARY aún no se ha completado satisfactoriamente para el nombre de LU especificado en este verbo o ha fallado la sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_INVALID_PROCESS

El proceso de sistema operativo que ha emitido este verbo no era el mismo proceso que ha emitido el verbo RUI_INIT o RUI_INIT_PRIMARY para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_NO_READ_TO_PURGE

El parámetro *lua_data_ptr* no contiene un puntero a un VCB RUI_READ o el verbo RUI_READ ha finalizado antes de que se emitiera el verbo RUI_PURGE.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si la sesión se ha iniciado utilizando RUI_INIT (no RUI_INIT_PRIMARY) y el código de retorno secundario no es LUA_RUI_LOGIC_ERROR, se puede reinicializar esta LU utilizando RUI_REINIT. Si no se reinicializa, se debe emitir RUI_TERM para que se pueda emitir RUI_INIT o RUI_INIT_PRIMARY para la misma LU.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

████████

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

Interacción con otros verbos

Este verbo sólo puede utilizarse cuando se ha emitido un verbo RUI_READ y todavía está pendiente de finalización (es decir, el código de retorno primario es IN_PROGRESS).

RUI_READ

El verbo RUI_READ recibe datos o información de estado enviada desde el sistema principal a la LU de la aplicación.

Puede especificar un determinado flujo de mensajes (LU normal, LU urgente, SSCP normal o SSCP urgente) del que se leerán los datos, o bien especificar más de un flujo de mensajes. Puede tener varios verbos RUI_READ pendientes, siempre y cuando no haya dos que especifiquen el mismo flujo.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Establézcalo en `sizeof(LUA_COMMON)`.

lua_opcode

LUA_OPCODE_RUI_READ

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Éste debe coincidir con el nombre de LU de una sesión LUA activa, tal como se devuelve en el verbo RUI_INIT o RUI_INIT_PRIMARY.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Éste debe coincidir con un ID de sesión devuelto en un verbo RUI_INIT o RUI_INIT_PRIMARY anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_max_length

Longitud del almacenamiento intermedio suministrado para recibir los datos.

lua_data_ptr

Puntero al almacenamiento intermedio suministrado para recibir los datos.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros *lua_flag1*

Establezca el parámetro *lua_flag1.nowait* en 1 si desea que el verbo RUI_READ vuelva tan pronto como sea posible, tanto si hay datos disponibles para leer como si no, o establézcalo en 0 (cero) si desea que el verbo espere hasta que reciba datos antes de volver.

Nota: El hecho de establecer el parámetro *lua_flag1.nowait* en 1 no significa que el verbo finalizará de forma síncrona. La biblioteca de LUA debe comunicarse con el nodo local para determinar si hay datos disponibles, y esto suele requerir el retorno asíncrono de un verbo para evitar el bloqueo de la aplicación. El parámetro significa que, si no hay datos disponibles de inmediato, el verbo asíncrono devolverá tan pronto como sea posible para indicarlo.

Establezca el parámetro *lua_flag1.bid_enable* en 1 para volver a activar el verbo RUI_BID más reciente (lo que equivale a volver a emitir RUI_BID con exactamente los mismos parámetros que antes), o establézcalo en 0 (cero) si no desea volver a activar RUI_BID. Si se vuelve a activar el verbo RUI_BID anterior, se vuelve a utilizar el VCB originalmente asignado para éste, de modo que este VCB no debe haberse liberado ni modificado. (Para ver más información, consulte “Interacción con otros verbos” en la página 89.)

Establezca uno o varios de los siguientes indicadores a 1 para indicar de qué flujo de mensajes se deben leer los datos:

lua_flag1.sscp_exp

lua_flag1.lu_exp

lua_flag1.sscp_norm

lua_flag1.lu_norm

Si se establece más de un indicador, se devolverán los datos disponibles cuya prioridad sea más alta. El orden de prioridad (de más alta a más baja) es: SSCP urgente, LU urgente, SSCP normal, LU normal. Se establecerá el indicador equivalente en el grupo *lua_flag2* para indicar de qué flujo se han leído los datos (consulte “Parámetros devueltos”).

La implementación de Communications Server para Linux de LUA no devuelve datos en el flujo acelerado de SSCP. La aplicación puede establecer el indicador *sscp_exp* para garantizar la compatibilidad con otras implementaciones de LUA, pero nunca se devolverán datos en este flujo.

Parámetros devueltos

LUA siempre devuelve los siguientes parámetros:

lua_flag2.async

Este parámetro tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 si ha finalizado de manera síncrona.

lua_flag2.bid_enable

Este parámetro tiene el valor 1 si se ha vuelto a activar correctamente un verbo RUI_BID, o el valor 0 si no se ha vuelto a activar.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta o datos truncados

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc
LUA_OK

Si el verbo finaliza de forma correcta, se devuelven los parámetros siguientes. También se devuelven si el verbo vuelve con datos truncados porque el parámetro *lua_data_length* suministrado era demasiado pequeño (consulte “Otras condiciones” en la página 87).

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_data_length
Longitud de los datos recibidos. LUA coloca los datos en el almacenamiento intermedio especificado por *lua_data_ptr*.

lua_th Información de la cabecera de transmisión (TH) del mensaje recibido.

lua_rh Información de la cabecera de petición/respuesta (RH) del mensaje recibido.

lua_message_type
Tipo de mensaje recibido; puede ser uno de los siguientes:

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIND
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_CLEAR
LUA_MESSAGE_TYPE_CRV
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SHUTD
LUA_MESSAGE_TYPE_SIGNAL
LUA_MESSAGE_TYPE_SDT
LUA_MESSAGE_TYPE_STSN
LUA_MESSAGE_TYPE_UNBIND

| |
|------------|
| AIX, LINUX |
|------------|

Los valores siguientes sólo se pueden devolver a una aplicación RUI primaria (la que ha iniciado la sesión utilizando RUI_INIT_PRIMARY):

LUA_MESSAGE_TYPE_INIT_SELF

LUA_MESSAGE_TYPE_NOTIFY

LUA_MESSAGE_TYPE_TERM_SELF



Parámetros de *lua_flag2*

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han recibido los datos:

lua_flag2.lu_exp

lua_flag2.sscp_norm

lua_flag2.lu_norm

La implementación de Communications Server para Linux de LUA no devuelve datos en el flujo rápido de SSCP y, por lo tanto, el indicador *sscp_exp* no se establecerá nunca (aunque lo puedan establecer otras implementaciones de LUA).

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque ha sido cancelado por otro verbo o por un error interno:

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

Los valores posibles son:

LUA_PURGED

Este verbo RUI_READ se ha cancelado mediante un verbo RUI_PURGE.

LUA_TERMINATED

Se ha emitido un verbo RUI_TERM mientras este verbo estaba pendiente.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

RUI_READ

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* contiene un valor que no es válido.

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_BID_ALREADY_ENABLED

Se ha establecido el parámetro *lua_flag1.bid_enable* para volver a activar un verbo RUI_BID, pero el verbo RUI_BID anterior todavía se estaba procesando.

LUA_DUPLICATE_READ_FLOW

Los indicadores de flujo del grupo *lua_flag1* especifican uno o varios flujos de sesión para los que ya había un verbo RUI_READ pendiente. No puede haber más de un verbo RUI_READ pendiente en un flujo de sesión.

LUA_INVALID_FLOW

No se ha establecido ninguno de los indicadores de flujo *lua_flag1*. Como mínimo uno de estos indicadores debe tener el valor 1 para indicar de qué flujo o flujos se debe leer.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_NO_PREVIOUS_BID_ENABLED

Se ha establecido el parámetro *lua_flag1.bid_enable* para volver a activar un verbo RUI_BID, pero no se ha podido activar ningún verbo RUI_BID anterior. (Para ver más información, consulte "Interacción con otros verbos" en la página 89.)

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

LUA_NO_RUI_SESSION

Un verbo RUI_INIT o RUI_INIT_PRIMARY aún no se ha completado satisfactoriamente para el nombre de LU especificado en este verbo o ha fallado la sesión.

Respuesta negativa enviada al sistema principal: El siguiente código de retorno primario indica uno de los dos casos siguientes, que puede distinguirse mediante el código de retorno secundario:

- Communications Server para Linux ha detectado un error en los datos recibidos del sistema principal. En lugar de pasar el mensaje recibido a la aplicación en un verbo RUI_READ, Communications Server para Linux elimina el mensaje (y el resto de la cadena si está en una cadena) y envía una respuesta negativa al sistema principal. LUA informa a la aplicación en un verbo RUI_READ o RUI_BID posterior de que se ha enviado una respuesta negativa.
- La aplicación LUA anteriormente había enviado una respuesta negativa a un mensaje en mitad de una cadena. Communications Server para Linux ha depurado los mensajes subsiguientes de esta cadena y ahora está informando a la aplicación que todos los mensajes de la cadena se han recibido y depurado.

lua_prim_rc

LUA_NEGATIVE_RSP

lua_sec_rc

Un código de retorno secundario distinto de cero contiene el código de detección enviado al sistema principal en la respuesta negativa. Esto indica que Communications Server para Linux ha detectado un error en los datos de sistema principal y ha enviado una respuesta negativa al sistema principal. Para ver información sobre cómo interpretar los valores de código de detección que pueden devolverse, consulte “Información de SNA” en la página 36.

Un código de retorno secundario de 0 (cero) indica que ahora, después de un RUI_WRITE anterior de una respuesta negativa a un mensaje en mitad de una cadena, Communications Server para Linux ha recibido y eliminado todos los mensajes de esta cadena.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_DATA_TRUNCATED

El parámetro *lua_data_length* es menor que la longitud real de los datos recibidos en el mensaje. Sólo se han devuelto *lua_data_length* bytes de datos al verbo; el resto de datos se ha eliminado. También se devuelven parámetros adicionales si se obtiene este código de retorno secundario; consulte “Ejecución correcta o datos truncados” en la página 84.

LUA_NO_DATA

El parámetro *lua_flag1.nowait* indica el retorno inmediato sin esperar a los datos, y no hay datos disponibles en el flujo o flujos de sesión especificados.

LUA_INVALID_PROCESS

El proceso de sistema operativo que ha emitido este verbo no era el mismo proceso que ha emitido el verbo RUI_INIT o RUI_INIT_PRIMARY para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si la sesión se ha iniciado utilizando RUI_INIT (no RUI_INIT_PRIMARY) y el código de retorno secundario no es LUA_RUI_LOGIC_ERROR, se puede reinicializar esta LU utilizando RUI_REINIT. Si no se reinicializa, se debe emitir RUI_TERM para que se pueda emitir RUI_INIT o RUI_INIT_PRIMARY para la misma LU.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

████████

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

Interacción con otros verbos

El verbo RUI_INIT o RUI_INIT_PRIMARY se tiene que haber completado satisfactoriamente para que se pueda emitir este verbo.

Si hay un verbo RUI_READ pendiente, puede emitir otro RUI_READ sólo si éste especifica un flujo (o flujos) de sesión diferente de los RUI_READ pendientes; no puede haber más de un RUI_READ pendiente para el mismo flujo de sesión.

El parámetro *lua_flag1.bid_enable* sólo puede utilizarse si se cumple lo siguiente:

- RUI_BID se ha emitido correctamente y ha finalizado
- El almacenamiento asignado para el verbo RUI_BID no se ha liberado ni modificado
- No hay ningún otro verbo RUI_BID pendiente

Si utiliza este parámetro para volver a activar un verbo RUI_BID anterior, debe seguir habiendo como mínimo un indicador de flujo de mensajes en RUI_READ, a fin de indicar el flujo o flujos en que la aplicación aceptará datos. Si los primeros datos que se van a recibir están en un flujo aceptado por el verbo RUI_READ, se devolverá RUI_READ con estos datos, pero no habrá retorno de RUI_BID. De lo contrario, se devolverá RUI_BID para indicar que hay datos que se deben leer (como RUI_BID acepta datos en todos los flujos, siempre aceptará los datos si RUI_READ no lo hace). En este caso, la aplicación debe emitir otro RUI_READ en el flujo apropiado para obtener los datos.

Si desea utilizar RUI_BID para gestionar datos en todos los flujos, en lugar de tener los datos en un determinado flujo gestionado preferentemente por RUI_READ en lugar de por RUI_BID, deberá volver a emitir RUI_BID explícitamente en vez de utilizar RUI_READ para volver a activar el verbo RUI_BID anterior.

Uso y restricciones

Si los datos recibidos superan la longitud máxima especificada en el parámetro *lua_max_length*, se truncarán; sólo se devolverán *lua_max_length* bytes de datos. También se devolverán los códigos de retorno primario y secundario LUA_UNSUCCESSFUL y LUA_DATA_TRUNCATED.

Una vez que se ha leído un mensaje utilizando el verbo RUI_READ, éste se elimina de la cola de mensajes de entrada y no puede volver a accederse a dicho mensaje. (El verbo RUI_BID puede utilizarse para una lectura "no destructiva"; la

RUI_READ

aplicación puede utilizarlo para consultar el tipo de datos disponibles, pero los datos permanecen en la cola de entrada y no es necesario utilizarlos inmediatamente.)

El ritmo puede utilizarse en la semisesión primaria a secundaria (esto se especifica en la configuración del sistema principal) a fin de evitar que la aplicación LUA se vea desbordada por un número excesivo de mensajes. Si la aplicación LUA es lenta al leer mensajes, Communications Server para Linux retarda el envío de las respuestas de ritmo al sistema principal a fin de reducir la velocidad del mismo.

RUI_REINIT

AIX, LINUX

El verbo RUI_REINIT restablece la sesión de SSCP-LU después de una anomalía de la sesión. Lo utilizan las aplicaciones que estaban utilizando una LU desde una agrupación y que necesitan asegurarse de que acceden a la misma LU a fin de continuar con el proceso. (Normalmente, una aplicación se recupera de una anomalía de sesión emitiendo RUI_TERM seguido de un segundo RUI_INIT; no obstante, si la aplicación utilizaba una LU de una agrupación, el segundo RUI_INIT no obtendrá necesariamente la misma LU que el original.)

Este verbo no se puede utilizar para reiniciar una sesión de RUI primaria (que se ha iniciado utilizando RUI_INIT_PRIMARY).

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA. Establézcalo en `sizeof(LUA_COMMON)`.

lua_opcode

LUA_OPCODE_RUI_REINIT

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU que se estaba utilizando en la sesión que ha fallado. Debe coincidir con el nombre devuelto en el verbo RUI_INIT original (no necesariamente el mismo que el nombre proporcionado al verbo).

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un identificador de sesión devuelto en el verbo RUI_INIT anterior de la sesión que ha fallado.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_post_handle

Puntero a una rutina de devolución de llamada. Si el verbo se completa de forma asíncrona, LUA llamará a esta rutina para indicar la finalización del verbo. Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 si ha finalizado de manera síncrona. (RUI_REINIT siempre finaliza de manera asíncrona, a menos que devuelva un error como, por ejemplo, LUA_PARAMETER_CHECK.)

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve el parámetro siguiente:

lua_prim_rc

LUA_OK

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo RUI_TERM antes de que finalizara RUI_REINIT.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

Los valores posibles son:

LUA_NO_RUI_SESSION

Un verbo RUI_INIT no se ha completado anteriormente de forma correcta para el nombre de LU o identificador de sesión.

LUA_DUPLICATE_RUI_INIT

En este momento se está procesando un verbo RUI_REINIT para esta sesión.

LUA_REINIT_INVALID

En esta sesión no se ha producido ninguna anomalía de sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

LUA_INVALID_PROCESS

El verbo RUI_INIT original se ha emitido desde un proceso de sistema operativo diferente.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

Este código de retorno indica una de las siguientes condiciones:

- El software Remote API Client no se ha iniciado. Inicie Remote API Client antes de ejecutar la aplicación.
- No hay nodos de Communications Server para Linux activos. Para poder utilizar los verbos LUA, debe haberse iniciado el nodo local propietario de la LU solicitada o un nodo local propietario de una o varias LU de la agrupación de LU solicitada. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si el código de retorno secundario no es `LUA_RUI_LOGIC_ERROR`, esta LU puede reiniciarse utilizando un verbo `RUI_REINIT`. Si no se reinicializa, debe emitirse un verbo `RUI_TERM` para que se pueda emitir un verbo `RUI_INIT` para la misma LU.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

La sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

Este verbo sólo puede emitirse si ha vuelto un verbo LUA con un código de retorno primario `LUA_SESSION_FAILURE` y con un código de retorno secundario distinto de `LUA_RUI_LOGIC_ERROR`.

Hasta que el verbo haya finalizado correctamente, el único otro verbo LUA que puede emitirse para esta sesión es `RUI_TERM` (que finalizará un `RUI_REINIT` pendiente).

Uso y restricciones

El verbo `RUI_REINIT` finaliza cuando se recibe un mensaje `ACTLU` del sistema principal. Si es necesario, el verbo espera de manera indefinida. Si ya se ha recibido un mensaje `ACTLU` antes del verbo `RUI_REINIT`, el verbo vuelve inmediatamente con el código de retorno primario `LUA_OK`.

Una vez que el verbo `RUI_REINIT` ha finalizado correctamente, esta sesión utiliza la LU para la que se ha iniciado la sesión. Ninguna otra sesión LUA (de esta u otra

RUI_REINIT

aplicación) puede utilizar la LU hasta que se emite el verbo RUI_TERM o hasta que se recibe el código de retorno primario LUA_SESSION_FAILURE.

Si el código de retorno secundario no es LUA_RUI_LOGIC_ERROR, esta LU puede reiniciarse utilizando un verbo RUI_REINIT. Si no se reinicializa, debe emitirse un verbo RUI_TERM para que se pueda emitir un verbo RUI_INIT para la misma LU.

El identificador de la sesión reiniciada es el mismo que el identificador de sesión de antes de que se produjera la anomalía. A diferencia de RUI_INIT, RUI_REINIT no devuelve este identificador de sesión; la aplicación debe utilizar el identificador de sesión devuelto al verbo RUI_INIT original o acceder a la sesión utilizando el nombre de la LU.



RUI_TERM

El verbo RUI_TERM finaliza la sesión de LU y la sesión de SSCP para una determinada LU.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Establézcalo en sizeof(LUA_COMMON).

lua_opcode

LUA_OPCODE_RUI_TERM

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa, como se ha devuelto en el verbo RUI_INIT o RUI_INIT_PRIMARY (o el nombre de LU que se ha especificado en un verbo RUI_INIT, RUI_INIT_PRIMARY o RUI_REINIT pendiente).

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Éste debe coincidir con un ID de sesión devuelto en un verbo RUI_INIT o RUI_INIT_PRIMARY anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

**LUA_RESERVED_FIELD_NOT_ZERO**

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

LUA_NO_RUI_SESSION

No hay ninguna sesión LUA con el nombre de LU especificado en este verbo, o la sesión ha fallado.

Si el verbo RUI_TERM se ha emitido para cancelar un verbo RUI_INIT, RUI_INIT_PRIMARY o RUI_REINIT pendiente, utilizando el parámetro *lua_luname* proporcionado al verbo pendiente, este código de retorno puede indicar que RUI_INIT, RUI_INIT_PRIMARY o RUI_REINIT se ha completado antes de que se procesara este verbo. Puede que el verbo no se haya completado correctamente (y, por lo tanto, no hay ninguna sesión), o que RUI_INIT haya finalizado correctamente utilizando una LU diferente de la agrupación especificada por *lua_luname* (y, por lo tanto, no hay ninguna sesión para el nombre de LU especificado).

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_COMMAND_COUNT_ERROR

Ya había un RUI_TERM pendiente cuando se ha emitido el verbo.

LUA_INVALID_PROCESS

El proceso de sistema operativo que ha emitido este verbo no era el mismo proceso que ha emitido el verbo RUI_INIT o RUI_INIT_PRIMARY para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

██████████

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

lua_prim_rc

RUI_TERM

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

Interacción con otros verbos

Este verbo se puede emitir en cualquier momento después de que se haya emitido el verbo RUI_INIT, RUI_INIT_PRIMARY o RUI_REINIT (tanto si se ha completado como si no lo ha hecho).

Si hay otro verbo LUA pendiente cuando se emite RUI_TERM, no se realiza ningún otro proceso en el verbo pendiente y éste devolverá con un código de retorno primario LUA_CANCELLED.

Una vez finalizado este verbo, no podrá emitirse ningún otro verbo LUA para esta sesión.

AIX, LINUX

Si la sesión se ha iniciado utilizando RUI_INIT_PRIMARY, Communications Server para Linux termina la sesión enviando DACTLU a la LU en sentido descendente. RUI_TERM no espera la respuesta de DACTLU para volver. Para empezar una nueva sesión con la LU en sentido descendente, la aplicación puede volver a emitir RUI_INIT_PRIMARY tan pronto como RUI_TERM ha finalizado; sin embargo, Communications Server para Linux no podrá procesar este RUI_INIT_PRIMARY hasta que haya recibido la respuesta DACTLU y, por consiguiente, es posible que RUI_INIT_PRIMARY tarde algún tiempo en completarse.



RUI_WRITE

El verbo RUI_WRITE envía una unidad de petición o respuesta SNA desde la aplicación LUA al sistema principal a través de la sesión de LU o de la sesión de SSCP.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_RUI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Establézcalo en sizeof(LUA_COMMON).

lua_opcode

LUA_OPCODE_RUI_WRITE

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Éste debe coincidir con el nombre de LU de una sesión LUA activa, tal como se devuelve en el verbo RUI_INIT o RUI_INIT_PRIMARY.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Éste debe coincidir con un ID de sesión devuelto en un verbo RUI_INIT o RUI_INIT_PRIMARY anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_data_length

Longitud de los datos proporcionados. Cuando se envían datos en el flujo LU normal, la longitud máxima es la especificada en el BIND recibido del sistema principal; para todos los demás flujos la longitud máxima es de 256 bytes.

Cuando se envía una respuesta afirmativa, este parámetro normalmente tiene el valor 0 (cero). LUA finalizará la respuesta en función del número de secuencia suministrado. En el caso de una respuesta afirmativa a un BIND o STSN, se permite una respuesta ampliada, de modo que puede utilizarse un valor distinto de cero.

Cuando se envía una respuesta negativa, establezca este parámetro a la longitud del código de detección SNA (cuatro bytes), que se suministra en el almacenamiento intermedio de datos.

lua_data_ptr

Puntero al almacenamiento intermedio que contiene los datos suministrados.

Para una petición o una respuesta afirmativa que requiere datos, el almacenamiento intermedio debe contener toda la RU. La longitud de la RU debe especificarse en *lua_data_length*.

Para una respuesta negativa, el almacenamiento intermedio debe contener el código de detección SNA.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función RUI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinRUI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

lua_th.snf

Necesario sólo cuando se envía una respuesta. El número de secuencia de la petición cuya respuesta es ésta.

lua_rh Cuando se envía una petición, la mayoría de los bits *lua_rh* deben establecerse para corresponder con la cabecera de petición (RH) del mensaje que se debe enviar. No establezca *lua_rh.pi* y *lua_rh.qri*; lo hará LUA.

Cuando se envía una respuesta, sólo se utilizan los dos bits *lua_rh* siguientes. Los demás deben tener el valor 0 (cero). Los bits *lua_rh* son:

lua_rh.rrl

Establézcalo a 1 para indicar una respuesta

lua_rh.ri

Establézcalo a 0 para una respuesta afirmativa, o a 1 para una respuesta negativa

Parámetros *lua_flag1*

Establezca uno de los siguientes indicadores a 1 para indicar en qué flujo de mensajes se enviarán los datos:

lua_flag1.lu_exp

lua_flag1.sscp_norm

lua_flag1.lu_norm

Únicamente uno de los indicadores se debe establecer en 1.

Communications Server para Linux no permite a las aplicaciones enviar datos en el flujo rápido de SSCP (el indicador *lua_flag1.sscp_exp*).

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_th TH finalizada del mensaje escrito, incluidos los campos rellenos por LUA. Quizás deba guardar el valor de *lua_th.snf* (el número de secuencia) para la correlación con las respuestas del sistema principal.

lua_rh RH finalizada del mensaje escrito, incluidos los campos rellenos por LUA.

Parámetros de *lua_flag2*

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han enviado los datos:

lua_flag2.lu_exp

lua_flag2.sscp_norm

lua_flag2.lu_norm

La implementación de Communications Server para Linux de LUA no permite a las aplicaciones enviar datos en el flujo rápido de SSCP y, por consiguiente, nunca establecerá el indicador *sscp_exp* (aunque es posible que lo establezcan otras implementaciones de LUA).

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

El verbo se ha cancelado porque se ha emitido un verbo RUI_TERM para esta sesión.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* contiene un valor que no es válido.

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_DUPLICATE_WRITE_FLOW

Un verbo RUI_WRITE ya estaba pendiente para el flujo de sesión especificado en este verbo (el flujo de sesión se especifica estableciendo uno de los indicadores de flujo *lua_flag1* a 1). No puede haber más de un verbo RUI_WRITE pendiente simultáneamente en un flujo de sesión.

LUA_INVALID_FLOW

Se ha establecido el indicador de flujo *lua_flag1.sscp_exp*, que indica

que el mensaje debe enviarse en el flujo rápido de SSCP. Communications Server para Linux no permite que las aplicaciones envíen datos en este flujo.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_MULTIPLE_WRITE_FLOWS

Más de uno de los indicadores de flujo *lua_flag1* tienen el valor 1. Únicamente uno de estos indicadores debe tener el valor 1 para indicar en qué flujo de sesión se van a enviar los datos.

LUA_REQUIRED_FIELD_MISSING

Este código de retorno indica uno de los siguientes casos:

- No se ha establecido ninguno de los indicadores de flujo *lua_flag1*. Únicamente uno de estos indicadores debe tener el valor 1.
- Se ha utilizado el verbo RUI_WRITE para enviar una respuesta, y la respuesta requería más datos de los suministrados.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
Los valores posibles son:

LUA_MODE_INCONSISTENCY

El mensaje SNA enviado en el verbo RUI_WRITE no era válido en este momento. Esto sucede al intentar enviar datos en la sesión de LU antes de que se haya establecido la sesión. Compruebe la secuencia de mensajes SNA enviada.

LUA_NO_RUI_SESSION

Un verbo RUI_INIT o RUI_INIT_PRIMARY aún no se ha completado satisfactoriamente para el nombre de LU especificado en este verbo o ha fallado la sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_FUNCTION_NOT_SUPPORTED

Este código de retorno indica uno de los siguientes casos:

- El bit *lua_rh.fi* (indicador de formato) tiene el valor 1, pero el primer byte de la RU suministrada no es un código de petición reconocido.
- El parámetro *lua_rh.ruc* (categoría RU) especificaba la categoría de Control de red (NC); Communications Server para Linux no permite que las aplicaciones envíen peticiones en esta categoría.

LUA_INVALID_PROCESS

El proceso de sistema operativo que ha emitido este verbo no era el mismo proceso que ha emitido el verbo RUI_INIT o RUI_INIT_PRIMARY para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_INVALID_SESSION_PARAMETERS

La aplicación ha utilizado RUI_WRITE para enviar una respuesta afirmativa a un mensaje BIND recibido del sistema principal. Sin embargo, el nodo de Communications Server para Linux no puede aceptar los parámetros BIND como se han especificado y ha enviado una respuesta negativa al sistema principal. Para obtener más información sobre los perfiles de BIND aceptados por Communications Server para Linux, consulte el apartado "Información de SNA" en la página 36.

LUA_RSP_CORRELATION_ERROR

Se ha utilizado RUI_WRITE para enviar una respuesta, pero el parámetro *lua_th.snf* (que indica el número de secuencia del mensaje recibido al que se responde) no contiene un valor válido.

LUA_RU_LENGTH_ERROR

El parámetro *lua_data_length* contiene un valor que no es válido. Cuando se envían datos en el flujo LU normal, la longitud máxima es la especificada en el BIND recibido del sistema principal; para todos los demás flujos la longitud máxima es de 256 bytes.

(cualquier otro valor)

Cualquier otro código de retorno secundario que aparezca aquí es un código de detección SNA que indica que los datos SNA suministrados no son válidos o no se han podido enviar. Para ver información sobre cómo interpretar los códigos de detección SNA que pueden devolverse, consulte "Información de SNA" en la página 36.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado.

Si la sesión se ha iniciado utilizando RUI_INIT (no RUI_INIT_PRIMARY) y el código de retorno secundario no es LUA_RUI_LOGIC_ERROR, se puede reinicializar esta LU utilizando RUI_REINIT. Si no se reinicializa, se debe emitir RUI_TERM para que se pueda emitir RUI_INIT o RUI_INIT_PRIMARY para la misma LU.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

■■■■■■

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha

iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

Interacción con otros verbos

Para poder emitir este verbo, se debe emitir satisfactoriamente el verbo RUI_INIT o RUI_INIT_PRIMARY.

Mientras hay un verbo RUI_WRITE pendiente, puede emitir un segundo verbo RUI_WRITE si éste especifica un flujo de sesión diferente al del verbo RUI_WRITE pendiente; es decir, no puede tener más de un verbo RUI_WRITE pendiente para el mismo flujo de sesión.

Se puede emitir el verbo RUI_WRITE en el flujo normal de SSCP en cualquier momento después de la ejecución satisfactoria de un verbo RUI_INIT o RUI_INIT_PRIMARY. Los verbos RUI_WRITE del flujo rápido de LU o del flujo normal de LU sólo están permitidos después de haberse recibido un mensaje BIND, y deben estar soportados por los protocolos especificados en el mensaje BIND.

Uso y restricciones

La finalización correcta de RUI_WRITE indica que el mensaje se ha puesto en cola correctamente para el enlace de datos; esto no indica necesariamente que el mensaje se haya enviado correctamente o que el sistema principal lo haya aceptado.

Se puede utilizar ritmo en la semisesión secundaria a primaria (esto se especifica en BIND), a fin de evitar que la aplicación LUA envíe más datos de los que puede manejar la LU de Communications Server para Linux o la LU de sistema principal. Si se da este caso, puede que LUA retrase un verbo RUI_WRITE en el flujo normal de LU y que tarde un poco en completarse.

Capítulo 5. Verbos SLI

Este capítulo contiene una descripción de cada verbo SLI LUA. Se proporciona la información siguiente para cada verbo:

- Propósito del verbo.
- Parámetros (campos del VCB) suministrados a LUA y devueltos por LUA. La descripción de cada parámetro incluye información sobre los valores válidos para el parámetro y toda la información adicional que sea necesaria.
- Interacción con otros verbos.
- Información adicional sobre el uso del verbo.

Para obtener detalles del bloque de control de verbos (VCB) que se utiliza para todos los verbos, consulte el Capítulo 3, “Estructura de VCB LUA”, en la página 47.

Se definen constantes simbólicas en los archivos de cabecera **lua_c.h** y **values_c.h** (sistema operativo Linux) o **winlua.h** (sistema operativo Windows) para muchos valores de parámetro. Para tener portabilidad, utilice la constante simbólica y no el valor numérico cuando establezca valores para los parámetros proporcionados o cuando pruebe los valores de los parámetros devueltos. El archivo **values_c.h** también incluye las definiciones de tipos de parámetro, como **AP_UINT16**, que se utilizan en los VCB LUA.

Los parámetros marcados como “reservados” siempre deben tener el valor 0 (cero).

SLI_BID

La aplicación utiliza el verbo SLI_BID para determinar cuándo un mensaje recibido está en espera de ser leído. Esto permite que la aplicación determine qué datos, si los hay, están disponibles antes de emitir el verbo SLI_RECEIVE.

Cuando hay un mensaje disponible, el verbo SLI_BID vuelve con detalles del flujo de mensajes en el que se ha recibido, el tipo de mensaje, la TH y RH del mensaje y hasta 12 bytes de datos del mensaje.

La diferencia principal entre SLI_BID y SLI_RECEIVE es que SLI_BID permite que la aplicación consulte los datos sin eliminarlos de la cola de mensajes de entrada, de modo que se puede acceder a dichos datos más adelante. El verbo SLI_RECEIVE elimina el mensaje de la cola, por lo que, una vez que la aplicación ha leído los datos, debe procesarlos.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

LUA_OPCODE_SLI_BID

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa (según lo devuelto en el verbo SLI_OPEN).

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un ID devuelto en un verbo SLI_OPEN anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_data_length

Número de bytes de los datos devueltos en el parámetro *lua_peek_data*; de 0 a 12.

lua_th TH del mensaje recibido.

lua_rh RH del mensaje recibido.

lua_message_type

Tipo de mensaje recibido, que puede ser uno de los siguientes:

LUA_MESSAGE_TYPE_LU_DATA

LUA_MESSAGE_TYPE_SSCP_DATA

LUA_MESSAGE_TYPE_RSP

LUA_MESSAGE_TYPE_BID

LUA_MESSAGE_TYPE_BIND

LUA_MESSAGE_TYPE_BIS

LUA_MESSAGE_TYPE_CANCEL

LUA_MESSAGE_TYPE_CHASE

LUA_MESSAGE_TYPE_LUSTAT_LU

LUA_MESSAGE_TYPE_LUSTAT_SSCP

LUA_MESSAGE_TYPE_QC

LUA_MESSAGE_TYPE_QEC

LUA_MESSAGE_TYPE_RELQ

LUA_MESSAGE_TYPE_RTR

LUA_MESSAGE_TYPE_SBI

LUA_MESSAGE_TYPE_SIGNAL

LUA_MESSAGE_TYPE_STSN

SLI utiliza las rutinas de ampliación de la interfaz de LUA de la aplicación para recibir y responder a las peticiones BIND y STSN.

lua_flag2

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han recibido los datos:

lua_flag2.sscp_exp

lua_flag2.lu_exp

lua_flag2.sscp_norm

lua_flag2.lu_norm

lua_peek_data

Los 12 primeros bytes de los datos del mensaje (o todos los datos del mensaje si son menos de 12 bytes)

Si *lua_rh.rrr* está desactivado (unidad de petición) y *lua_rh.sdi* está activado (datos de detección incluidos), esto indica que LUA ha convertido una unidad de petición enviada por el sistema principal en una petición de excepción (EXR). En este caso, los bytes 0–3 de *lua_peek_data* contienen los datos de detección asociados con la excepción y los bytes 4–6 contienen hasta los 3 primeros bytes de la unidad de petición original.

Ejecución correcta: información de estado

Si el verbo devuelve información de estado de LUA en lugar de datos, LUA devuelve los parámetros siguientes:

lua_prim_rc
LUA_STATUS

lua_sec_rc

LUA_READY

La sesión SLI está preparada para procesar mandatos adicionales. Este estado se utiliza después de haber informado de un estado LUA_NOT_READY anterior o después de haber finalizado un verbo SLI_CLOSE con *lua_prim_rc* establecido en LUA_CANCELLED y *lua_sec_rc* establecido en RECEIVED_UNBIND_HOLD o RECEIVED_UNBIND_NORMAL.

LUA_NOT_READY

La sesión SLI se ha suspendido temporalmente por uno de los motivos siguientes:

- Se ha recibido un mandato CLEAR. La sesión se reanuda cuando se recibe un mandato SDT.
- Se ha recibido un mandato UNBIND de tipo X'02' (BIND próximo). La sesión se suspende hasta que se reciben los mandatos BIND, CRV y STSN opcionales y SDT; se reanuda después del SDT. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo SLI_OPEN original.
- Se ha recibido un mandato UNBIND de tipo X'01' (normal) y el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED. La sesión se suspende hasta que se reciben los mandatos BIND, CRV y STSN opcionales y SDT; se reanuda después del SDT. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo SLI_OPEN original.

La aplicación debería emitir otro SLI_BID o SLI_RECEIVE para recibir el estado READY cuando la sesión se reanuda. Puede seguir emitiendo verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP aunque el estado de la sesión sea LUA_NOT_READY.

LUA_INIT_COMPLETE

La aplicación ha emitido SLI_OPEN con el tipo LUA_OPEN_TYPE_PRIM_SSCP y el verbo RUI_INIT subyacente ha finalizado ahora. La aplicación ahora puede emitir verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP.

LUA_SESSION_END_REQUESTED

El sistema principal ha enviado un mandato SHUTD para solicitar a la aplicación que cierre la sesión. La aplicación debería emitir SLI_CLOSE tan pronto como esté preparada para cerrar la sesión.

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo SLI_CLOSE mientras este verbo estaba pendiente.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_INVALID_LUNAME

No se ha encontrado la LU identificada por el parámetro *lua_luname* en ningún nodo activo. Compruebe que el nombre de LU o el nombre de agrupación de LU está definido en el archivo de configuración y que el nodo en el que está configurado se ha iniciado.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

LUA_NO_SLI_SESSION

Un verbo SLI_OPEN todavía no ha finalizado correctamente para el nombre de LU especificado en este verbo o ha fallado la sesión.

LUA_SLI_BID_PENDING

Se ha rechazado el verbo SLI_BID porque todavía había un verbo SLI_BID anterior pendiente para esta sesión. No puede haber más de un verbo SLI_BID pendiente para una misma sesión.

Respuesta negativa enviada al sistema principal: El código de retorno siguiente indica que Communications Server para Linux ha detectado un error en los datos recibidos del sistema principal. En lugar de pasar el mensaje recibido a la aplicación en un verbo SLI_RECEIVE, Communications Server para Linux elimina el mensaje (y el resto de la cadena si está en una cadena) y envía una respuesta negativa al sistema principal. LUA informa a la aplicación en un verbo SLI_RECEIVE o SLI_BID posterior de que se ha enviado una respuesta negativa.

lua_prim_rc
LUA_NEGATIVE_RSP

lua_sec_rc

El código de retorno secundario contiene el código de detección enviado al sistema principal en la respuesta negativa. Consulte "Información de SNA" en la página 36, para ver información sobre cómo interpretar los valores de código de detección que pueden devolverse.

Un código de retorno secundario de 0 (cero) indica que ahora, después de un SLI_SEND anterior de una respuesta negativa a un mensaje en mitad de una cadena, Communications Server para Linux ha recibido y eliminado todos los mensajes de esta cadena.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc

LUA_INVALID_PROCESS

El proceso del sistema operativo que ha emitido este verbo no es el mismo que el proceso que ha emitido el verbo SLI_OPEN para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RECEIVED_UNBIND

Este código de retorno indica que el sistema principal ha enviado un mandato UNBIND para finalizar la sesión. Este valor sólo se puede producir si el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED.

LUA_RUI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

WINDOWS

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

■■■■■

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

El verbo SLI_OPEN debe completarse correctamente para que se pueda emitir este verbo.

No puede haber más de un verbo SLI_BID pendiente simultáneamente para cada sesión.

Una vez que el verbo SLI_BID ha finalizado correctamente, puede volver a emitirse enviando el parámetro *lua_flag1.bid_enable* en otro verbo SLI_RECEIVE. Si el verbo se volverá a emitir de este modo, el programa de aplicación no debe liberar ni modificar el almacenamiento asociado con el registro de verbo SLI_BID.

Si llega un mensaje del sistema principal cuando hay un verbo SLI_RECEIVE y un verbo SLI_BID pendientes, SLI_RECEIVE finaliza y SLI_BID se deja en proceso.

Uso y restricciones

Cada mensaje que llega se anuncia una sola vez. Cuando el verbo SLI_BID ha indicado que hay datos en espera en un determinado flujo de sesión, la aplicación debe emitir el verbo SLI_RECEIVE para recibir los datos. Todos los verbos SLI_BID posteriores no informarán de la llegada de datos en ese flujo de sesión hasta que el mensaje que se ha anunciado se haya aceptado emitiendo un verbo SLI_RECEIVE.

Si hay más datos disponibles para más de un flujo de sesión, los datos del flujo con la prioridad más alta se devolverán a la aplicación. Las prioridades de flujo se indican a continuación (de la más alta a la más baja):

- SSCP rápido
- LU rápido
- SSCP normal
- LU normal

Una vez que se ha leído un mensaje utilizando el verbo SLI_RECEIVE, éste se elimina de la cola de mensajes de entrada y no puede volver a accederse a dicho mensaje. La aplicación puede utilizar SLI_BID como una lectura no destructiva para comprobar el tipo de datos disponibles y determinar cómo procesarlos, y después emitir un SLI_RECEIVE posterior para recopilar los datos. Sin embargo, si emite SLI_RECEIVE con varios indicadores *lua_flag1* establecidos para aceptar datos en más de un flujo, puede recibir un mensaje distinto del mensaje identificado en SLI_BID, si han llegado datos en un flujo de prioridad más alta entre los verbos SLI_BID y SLI_RECEIVE. Para asegurarse de que recibe el mismo mensaje identificado en SLI_BID, debería establecer los identificadores *lua_flag1* en SLI_RECEIVE para aceptar datos únicamente en el flujo identificado en la respuesta de SLI_BID.

El parámetro *lua_data_length* indica la longitud de los datos de *lua_peek_data*. Si es inferior a 12 (lo cual indica que el mensaje en espera tiene menos de 12 bytes), los demás bytes de *lua_peek_data* no están definidos y la aplicación no debe intentar examinarlos.

SLI_CLOSE

El verbo SLI_CLOSE finaliza la sesión de LU y la sesión de SSCP para una determinada LU.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

LUA_OPCODE_SLI_CLOSE

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa, según lo devuelto en el verbo SLI_OPEN (o el nombre de LU especificado en un verbo SLI_OPEN pendiente).

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un ID devuelto en un verbo SLI_OPEN anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.

■

Parámetros *lua_flag1*

Establezca el parámetro *lua_flag1.close_abend* en 1 si desea que la sesión se cierre inmediatamente o establézcalo en 0 (cero) si desea que SLI efectúe el intercambio normal de mensajes SNA con el sistema principal para cerrar

la sesión con el procedimiento normal. Para obtener más detalles sobre el proceso de cierre normal o anormal, consulte “Uso y restricciones” en la página 120.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc

LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_INVALID_LUNAME

No se ha encontrado la LU identificada por el parámetro *lua_luname* en ningún nodo activo. Compruebe que el nombre de LU o el nombre de agrupación de LU está definido en el archivo de configuración y que el nodo en el que está configurado se ha iniciado.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

Los valores posibles son:

LUA_CLOSE_PENDING

La aplicación ha emitido SLI_CLOSE (normal) cuando ya había un SLI_CLOSE (cierre normal o anormal) en proceso o ha emitido SLI_CLOSE (cierre anormal) cuando ya había un SLI_CLOSE (cierre anormal) en proceso. Un segundo SLI_CLOSE sólo es válido si es un SLI_CLOSE (cierre anormal) posterior a un SLI_CLOSE (normal) previo.

LUA_NO_SLI_SESSION

No hay ninguna sesión LUA con el nombre de LU especificado en este verbo, o ha fallado la sesión.

Si se ha emitido el verbo SLI_CLOSE para cancelar un verbo SLI_OPEN pendiente, utilizando el parámetro *lua_luname* suministrado al verbo pendiente, este código de retorno puede indicar que SLI_OPEN ha finalizado antes de que se procesara este verbo. Puede que el verbo no se haya completado correctamente (y, por lo tanto, no hay ninguna sesión) o que SLI_OPEN haya finalizado correctamente utilizando una LU diferente de la agrupación especificada por *lua_luname* (y, por lo tanto, no hay ninguna sesión para el nombre de LU especificado).

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque un mensaje enviado desde el sistema principal lo ha cancelado:

lua_prim_rc

LUA_CANCELLED

lua_sec_rc

Los valores posibles son:

LUA_RECEIVED_UNBIND_HOLD

Este verbo SLI_CLOSE lo ha cancelado un UNBIND de tipo 0x02 (UNBIND con BIND próximo) desde el sistema principal. La sesión no se ha cerrado; la aplicación debería emitir SLI_BID o SLI_RECEIVE para obtener información de estado. Cualquier rutina de ampliación del usuario que la aplicación especifique en el verbo SLI_OPEN se llamará otra vez cuando el sistema principal envíe el nuevo BIND.

LUA_RECEIVED_UNBIND_NORMAL

Este verbo SLI_CLOSE lo ha cancelado un UNBIND de tipo 0x01 (UNBIND normal) desde el sistema principal y el parámetro *lua_session_type* en el SLI_OPEN que ha iniciado la sesión se ha establecido en LUA_SESSION_TYPE_DEDICATED. La sesión no se ha cerrado; la aplicación debería emitir SLI_BID o SLI_RECEIVE para obtener información de estado. Cualquier rutina de ampliación del

usuario que la aplicación especifique en el verbo SLI_OPEN se llamará otra vez cuando el sistema principal envíe el nuevo BIND. Si la aplicación desea finalizar la sesión sin esperar un nuevo BIND, debe emitir SLI_CLOSE (cierre anormal).

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

LUA_INVALID_PROCESS

El proceso del sistema operativo que ha emitido este verbo no es el mismo que el proceso que ha emitido el verbo SLI_OPEN para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_NAU_INOPERATIVE

Un componente SNA necesario (como la LU de LUA) no está activo o tiene un estado anormal.

LUA_NO_SESSION

La sesión SNA con la LU remota no está activa.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_NEGATIVE_RSP_CHASE

Este código de retorno indica que la sesión LUA se ha cerrado porque SLI ha recibido una respuesta negativa a un mandato CHASE.

LUA_NEGATIVE_RSP_SHUTD

Este código de retorno indica que la sesión LUA se ha cerrado porque SLI ha recibido una respuesta negativa a un mandato SHUTD.

LUA_NEGATIVE_RSP_RSHUTD

Este código de retorno indica que la sesión LUA se ha cerrado porque SLI ha recibido una respuesta negativa a un mandato RSHUTD.

LUA_RECEIVED_UNBIND

Este código de retorno indica que el sistema principal ha enviado un mandato UNBIND para finalizar la sesión. Este valor sólo se puede producir si el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED.

LUA_UNEXPECTED_SNA_SEQUENCE

Este código de retorno indica que la sesión LUA se ha cerrado porque SLI ha recibido un mensaje SNA inesperado desde el sistema principal.

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

Interacción con otros verbos

Este verbo se puede emitir en cualquier momento después de que se haya emitido el verbo SLI_OPEN. Si SLI_OPEN todavía no ha finalizado y la aplicación desea cancelarlo, deberá hacerlo emitiendo SLI_CLOSE con *lua_flag1.close_abend* establecido en 1 (para indicar un cierre anormal).

Mientras un SLI_CLOSE (normal) está pendiente, la aplicación puede emitir un SLI_CLOSE (cierre anormal) si determina que necesita finalizar la sesión rápidamente sin esperar a que se realice el proceso de cierre normal.

Si hay otro verbo LUA pendiente cuando se emite SLI_CLOSE, no se realiza ningún otro proceso en el verbo pendiente, que devolverá el código de retorno primario LUA_CANCELLED.

Una vez finalizado este verbo, no podrá emitirse ningún otro verbo LUA para esta sesión. La aplicación puede emitir SLI_OPEN para la misma LU o para una LU distinta, para iniciar una nueva sesión.

Uso y restricciones

El proceso de cierre de sesión lo puede iniciar el sistema principal (cierre de inicio primario) o la aplicación LUA (cierre de inicio secundario), tal como se describe a continuación. En ambos casos la aplicación normalmente establece *lua_flag1.close_abend* en 0 (cero), para indicar un cierre normal en el que LUA y el sistema principal intercambian la secuencia habitual de mensajes al final de la sesión.

Cierre de inicio primario

El sistema principal inicia el cierre del proceso enviando un mandato SHUTD, que se devuelve a la aplicación como un valor de estado LUA_SESSION_END_REQUESTED en un verbo SLI_BID o SLI_RECEIVE.

Cuando la aplicación está preparada para cerrar la sesión, responde emitiendo SLI_CLOSE. Esto genera la siguiente secuencia de mensajes entre LUA y el sistema principal.

- LUA envía CHASE al sistema principal y recibe la respuesta.
- LUA envía SHUTC (Shutdown Complete) al sistema principal y recibe la respuesta.
- Opcionalmente, el sistema principal envía CLEAR; LUA lo recibe y envía la respuesta.
- El sistema principal envía UNBIND; LUA lo recibe y envía la respuesta.
- LUA detiene la sesión RUI y se devuelve el verbo SLI_CLOSE.

Cierre de inicio secundario

La aplicación inicia el cierre del proceso emitiendo SLI_CLOSE. Esto genera la siguiente secuencia de mensajes entre LUA y el sistema principal.

- LUA envía RSHUTD al sistema principal y recibe la respuesta.
- Opcionalmente, el sistema principal envía CLEAR; LUA lo recibe y envía la respuesta.
- El sistema principal envía UNBIND; LUA lo recibe y envía la respuesta.
- LUA detiene la sesión RUI y se devuelve el verbo SLI_CLOSE.

Mientras está en proceso un SLI_CLOSE (normal), el sistema principal lo puede interrumpir enviando uno de los mensajes siguientes:

- UNBIND de tipo 0x02 (UNBIND con BIND próximo)
- UNBIND de tipo 0x01 (UNBIND normal), si el parámetro *lua_session_type* en el SLI_OPEN que ha iniciado la sesión se ha establecido en LUA_SESSION_TYPE_DEDICATED

En los dos casos, el verbo SLI_CLOSE devuelve el código de retorno primario CANCELLED. La sesión no se ha cerrado; la aplicación debería emitir SLI_BID o

SLI_RECEIVE para obtener información de estado. Cualquier rutina de ampliación del usuario que la aplicación especifique en el verbo SLI_OPEN se llamará otra vez cuando el sistema principal envíe el nuevo BIND.

Si la aplicación debe finalizar la sesión rápidamente sin esperar la secuencia de mensajes habitual o debe cerrar una sesión dedicada sin esperar un nuevo BIND después de que el sistema principal haya enviado UNBIND (normal), lo hace emitiendo SLI_CLOSE con *lua_flag1.close_abend* establecido en 1. Esta acción finaliza la sesión SLI; LUA realizará todo el proceso de limpieza que sea necesario para informar al sistema principal de que la sesión ha finalizado.

Antes de emitir SLI_CLOSE (normal), con *lua_flag1.close_abend* establecido en 0 (cero), la aplicación debe asegurarse de que ha recibido todos los mensajes pendientes desde el sistema principal y ha enviado todas las respuestas necesarias. Si no se ha enviado una respuesta necesaria, LUA cambia automáticamente el tipo de cierre y realiza un CLOSE (cierre anormal) del proceso tal como se describe más arriba.

SLI_OPEN

El verbo SLI_OPEN establece la sesión SNA para una LU determinada o para la LU utilizada menos recientemente de una agrupación de LU determinada.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

LUA_OPCODE_SLI_OPEN

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU o agrupación de LU para la que desea iniciar la sesión. Debe coincidir con el nombre de una LU de tipo 0–3, o de una agrupación de LU, configurada para Communications Server para Linux. El nombre se utiliza de la siguiente manera:

- Si el nombre es el nombre de una LU que no está en una agrupación, Communications Server para Linux intenta iniciar la sesión utilizando esta LU. Una aplicación puede iniciar varias sesiones utilizando varios verbos SLI_OPEN con una LU diferente para cada verbo; no puede iniciar más de una sesión para la misma LU.
- Si el nombre es el nombre de una agrupación de LU, o el nombre de una LU de una agrupación, Communications Server para Linux intenta iniciar la sesión utilizando la LU mencionada si está disponible o, de lo contrario, la LU utilizada menos recientemente de la agrupación. Una aplicación puede iniciar varias sesiones utilizando la misma agrupación; Communications Server para Linux asignará una LU diferente de la

SLI_OPEN

agrupación para cada sesión. El nombre de la LU real utilizada para la sesión es un parámetro devuelto en el verbo SLI_OPEN.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_data_length

La longitud de los datos de LOGON o INITSELF sin formatear suministrados en el parámetro *lua_data_ptr* o cero si no es necesario suministrar datos.

lua_data_ptr

Un puntero hacia el mensaje, si existe alguno, que debe enviarse al sistema principal para iniciar la sesión. Esto depende del parámetro *lua_init_type*, tal como se indica a continuación.

- Si *lua_init_type* es `LUA_INIT_TYPE_SEC_IS`, la aplicación debe proporcionar una unidad de petición INITSELF que contenga la información de usuario necesaria como, por ejemplo, el nombre de modalidad y el nombre de PLU.
- Si *lua_init_type* es `LUA_INIT_TYPE_SEC_LOG`, la aplicación debe proporcionar un mensaje LOGON sin formatear para enviarlo en el flujo normal de SSCP.
- Si *lua_init_type* es `LUA_INIT_TYPE_PRIM` o `LUA_INIT_TYPE_PRIM_SSCP`, este parámetro no se utiliza y la aplicación debe proporcionar un puntero nulo.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona. (Si el verbo no pasa las comprobaciones iniciales de LUA y el punto de entrada SLI devuelve cero, LU no llamará a esta rutina.)

WINDOWS

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

lua_encr_decr_option

Este parámetro está reservado y debe establecerse en cero.

lua_init_type

Especifica cómo LUA debe iniciar la sesión. Los valores posibles son:

LUA_INIT_TYPE_SEC_IS

Inicio secundario: se envía el mensaje INITSELF de la aplicación (indicado por *lua_data_ptr*) al sistema principal.

LUA_INIT_TYPE_SEC_LOG

Inicio secundario: se envía el mensaje LOGON sin formatear de la aplicación (indicado por *lua_data_ptr*) al sistema principal.

LUA_INIT_TYPE_PRIM

Inicio primario: se espera un BIND desde el sistema principal.

LUA_INIT_TYPE_PRIM_SSCP

Inicio primario con acceso a SSCP: permite que la aplicación emita verbos SLI_SEND y SLI_RECEIVE en el flujo normal de SSCP, de modo que pueda proporcionar sus propios mensajes INITSELF o LOGON y recibir las respuestas. Después de emitir SLI_OPEN, la aplicación puede emitir SLI_BID o SLI_RECEIVE para obtener la indicación de estado INIT_COMPLETE y, a continuación, puede utilizar SLI_SEND y SLI_RECEIVE para enviar mensajes INITSELF o LOGON y recibir las respuestas.

lua_session_type

Especifica cómo LUA debe procesar un UNBIND de tipo X'01' (normal). Los valores posibles son:

LUA_SESSION_TYPE_NORMAL

Se envía una respuesta afirmativa y se emite RUI_TERM de modo que se envíe un NOTIFY(inhabilitado) a SSCP. El flujo SSCP-LU está inhabilitado.

LUA_SESSION_TYPE_DEDICATED

Se envía una respuesta afirmativa y se suspende la sesión SLI hasta que se reciben los mandatos BIND, CRV y STSN opcionales, y SDT. NOTIFY(inhabilitado) no se envía a SSCP. En este caso la aplicación puede finalizar la sesión suspendida, sin esperar un nuevo BIND desde el sistema principal, emitiendo SLI_CLOSE (cierre anormal).

lua_wait

Tiempo de espera (en segundos) para reintentar un inicio secundario de sesión. Este parámetro se ignora si *lua_init_type* es LUA_INIT_TYPE_PRIM o LUA_INIT_TYPE_PRIM_SSCP.

LUA reintenta el inicio de sesión después de este tiempo de espera (volviendo a enviar el mensaje INITSELF o LOGON de la aplicación) si el sistema principal responde al intento inicial con uno de los mensajes siguientes.

- Una respuesta negativa a INITSELF o LOGON con un código de retorno secundario RESOURCE_NOT_AVAILABLE, SESSION_LIMIT_EXCEEDED, SSCP_LU_SESS_NOT_ACTIVE o SESSION_SERVICE_PATH_ERROR.
- Un mensaje NSPE (error de procedimiento de servicios de red).
- Un mandato NOTIFY, que indica un error de procedimiento.

Si este parámetro se establece en cero, LUA no reintenta el inicio de sesión.

lua_open_extension

Información sobre las rutinas de ampliación de SLI_OPEN, si existe alguna. Este parámetro es una matriz de estructuras, cada una de las cuales contiene información sobre una rutina de ampliación específica.

La aplicación puede especificar de 0 a 3 rutinas de ampliación, cada una de las cuales identifica la rutina de la aplicación para manejar un mensaje SNA específico durante la inicialización de la sesión (tal como indica el parámetro *lua_routine_type*). Deben especificarse en elementos consecutivos

en la matriz, empezando por el primero; todas las entradas proporcionadas deben finalizar con uno en el cual *lua_open_extension.lua_routine_type* esté establecido en `LUA_ROUTINE_TYPE_END`, para indicar el final de la lista.

lua_open_extension.lua_routine_type

Tipo de rutina de ampliación. Los valores posibles son:

LUA_ROUTINE_TYPE_BIND

Rutina para comprobar y responder a un mensaje BIND desde el sistema principal.

LUA_ROUTINE_TYPE_SDT

Rutina para comprobar y responder a un mensaje SDT desde el sistema principal.

LUA_ROUTINE_TYPE_STSN

Rutina para comprobar y responder a un mensaje STSN desde el sistema principal.

LUA_ROUTINE_TYPE_END

Este valor indica el final de la lista de rutinas de ampliación. Debe utilizarse en el elemento de la matriz que sigue inmediatamente a las otras rutinas (o en el primer elemento de la matriz si la aplicación no especifica ninguna rutina de ampliación).

AIX, LINUX

lua_open_extension.lua_routine_ptr

Puntero hacia el punto de entrada de la rutina de ampliación. Este parámetro no se utiliza en la última entrada de la matriz, en la cual *lua_open_extension.lua_routine_type* está establecido en `LUA_ROUTINE_TYPE_END`.

LUA llama a este punto de entrada con el verbo `SLI_BIND_ROUTINE`, `SLI_SDT_ROUTINE` o `SLI_STSN_ROUTINE`, según el valor del parámetro *lua_routine_type*.

WINDOWS

lua_open_extension.lua_module_name

Nombre de la DLL que contiene el módulo de ampliación. Este parámetro no se utiliza en la última entrada de la matriz, en la cual *lua_open_extension.lua_routine_type* está establecido en `LUA_ROUTINE_TYPE_END`.

lua_open_extension.lua_procedure_name

Nombre de procedimiento a llamar dentro de la DDL del módulo de ampliación. Este parámetro no se utiliza en la última entrada de la matriz, en la cual *lua_open_extension.lua_routine_type* está establecido en `LUA_ROUTINE_TYPE_END`.

LUA llama a este punto de entrada con el verbo `SLI_BIND_ROUTINE`, `SLI_SDT_ROUTINE` o `SLI_STSN_ROUTINE`, según el valor del parámetro *lua_routine_type*.



lua_ending_delim

La interfaz SLI de Communications Server para Linux no utiliza este parámetro; se proporciona por compatibilidad con aplicaciones escritas originalmente para otras implementaciones de SLI.

Valor de retorno desde el punto de entrada SLI

El verbo SLI_OPEN es el único verbo para el cual el punto de entrada SLI devuelve un valor.

- Si el verbo no pasa las comprobaciones iniciales de LUA (por ejemplo, porque la aplicación ha proporcionado parámetros incorrectos), la llamada de función SLI devuelve un valor cero para indicarlo. La aplicación deberá comprobar los parámetros *lua_prim_rc* y *lua_sec_rc* para determinar la causa de la anomalía. Communications Server para Linux no llama a la rutina de devolución de llamada proporcionada por la aplicación.
- Si las comprobaciones iniciales son satisfactorias, la llamada de función SLI devuelve un valor distinto de cero que representa el ID de sesión de la sesión nueva. Si *lua_init_type* se ha establecido en LUA_INIT_TYPE_PRIM_SSCP, la aplicación puede utilizar este ID de sesión para verbos SLI_BID o SLI_RECEIVE posteriores en el flujo normal de SSCP (para recibir el indicador de estado INIT_COMPLETE) y, a continuación, para verbos SLI_SEND y SLI_RECEIVE en este flujo.

A continuación, Communications Server para Linux utiliza la rutina de devolución de llamada proporcionada por la aplicación del mismo modo que para otros verbos SLI.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo se ejecuta de forma correcta, LUA devuelve los parámetros siguientes.

lua_prim_rc

LUA_OK

lua_sid Identificador de sesión para la nueva sesión. Es igual que el valor de retorno desde el punto de entrada SLI para este verbo y se puede utilizar en verbos posteriores para identificar esta sesión.

lua_luname

Nombre de la LU utilizada por la nueva sesión. Si el nombre de LU en los parámetros de petición especificaba una agrupación de LU, Communications Server para Linux utiliza este parámetro para devolver el nombre de la LU real asignada a la sesión. Los verbos posteriores deben utilizar este nombre devuelto (no el nombre especificado en los parámetros de petición) para identificar la sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo SLI_CLOSE antes de que finalizara SLI_OPEN.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_DATA_LENGTH_ERROR

El parámetro *lua_init_type* especifica una sesión de inicio secundario, pero la aplicación no ha proporcionado los datos necesarios para enviarlos al sistema principal.

LUA_INVALID_LUNAME

No se ha encontrado la LU identificada por el parámetro *lua_luname* en ningún nodo activo. Compruebe que el nombre de LU o el nombre de agrupación de LU está definido en el archivo de configuración y que el nodo en el que está configurado se ha iniciado.

LUA_INVALID_OPEN_DATA

El parámetro *lua_init_type* se ha establecido en LUA_INIT_TYPE_SEC_IS, pero el almacenamiento intermedio de datos indicado por *lua_data_ptr* no contenía ningún mandato INITSELF válido.

LUA_INVALID_OPEN_INIT_TYPE

El parámetro *lua_init_type* no se ha establecido en un valor válido.

LUA_INVALID_OPEN_ROUTINE_TYPE

El parámetro *lua_routine_type* no se ha establecido en un valor válido.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.



LUA_INVALID_SESSION_TYPE

El parámetro *lua_session_type* no se ha establecido en un valor válido.

LUA_INVALID_SLI_ENCR_OPTION

El parámetro *lua_encr_decr_option* no se ha establecido en un valor válido. Para Communications Server para Linux, este parámetro se debe establecer en 0 (cero).

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* contiene un valor que no es válido.

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc

LUA_STATE_CHECK

lua_sec_rc

LUA_DUPLICATE_RUI_INIT

En este momento se está procesando un verbo SLI_OPEN para esta sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_COMMAND_COUNT_ERROR

El verbo especifica el nombre de una agrupación de LU o el nombre de una LU de una agrupación, pero ya se están utilizando todas las LU de la agrupación.

LUA_INVALID_PROCESS

Otro proceso está utilizando la LU especificada por el parámetro *lua_luname*.

LUA_LINK_NOT_STARTED

La conexión con el sistema principal no se ha iniciado; ninguno de los enlaces que puede utilizar está activo.

LUA_SESSION_ALREADY_OPEN

La aplicación ha proporcionado un nombre de LU para el cual ya se ha iniciado una sesión.

LUA_NAU_INOPERATIVE

Un componente SNA necesario (como la LU de LUA) no está activo o tiene un estado anormal.

LUA_NO_SESSION

La sesión SNA con la LU remota no está activa.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

(cualquier otro valor)

Cualquier otro código de retorno secundario que aparezca aquí es un código de detección SNA. Para ver información sobre cómo interpretar los códigos de detección SNA que pueden devolverse, consulte "Información de SNA" en la página 36.

Los siguientes valores de código de detección son específicos de Communications Server para Linux y pueden indicar discrepancias entre la configuración de Communications Server para Linux y la configuración de sistema principal:

0x10020000

El sistema principal no ha enviado un mensaje de activación de unidad física (ACTPU) para la PU propietaria de la LU solicitada.

0x10110000

El sistema principal no ha enviado un mensaje ACTLU para la LU solicitada. Normalmente esto indica que la LU no está configurada en el sistema principal.

0x10120000

El sistema principal no ha enviado un mensaje ACTLU para la LU solicitada. El sistema principal soporta DDDL (definición dinámica de LU dependientes), pero el proceso DDDL para esta LU ha fallado.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

Este código de retorno indica una de las siguientes condiciones:

- El software Remote API Client no se ha iniciado. Inicie Remote API Client antes de ejecutar la aplicación.
- No hay nodos de Communications Server para Linux activos. Para poder utilizar los verbos LUA, debe haberse iniciado el nodo local propietario de la LU solicitada o un nodo local propietario de una o varias LU de la agrupación de LU solicitada. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

La sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

El verbo SLI_OPEN debe ser el primer verbo LUA emitido para la sesión.

Hasta que el verbo haya finalizado correctamente, los únicos otros verbos LUA que se pueden emitir para esta sesión son:

- SLI_CLOSE con *lua_flag1.close_abend* establecido en 1 (para indicar un cierre anormal), que cancelará el SLI_OPEN pendiente.

SLI_OPEN

- Si *lua_init_type* se ha establecido en `LUA_INIT_TYPE_PRIM_SSCP`:
 - `SLI_BID` o `SLI_RECEIVE` para obtener la indicación de estado `INIT_COMPLETE`
 - `SLI_SEND` y `SLI_RECEIVE` para datos de flujo normal de SSCP, para enviar mensajes `INITSELF` o `LOGON` y recibir las respuestas.

Todos los demás verbos emitidos en esta sesión deben identificar la sesión utilizando uno de los parámetros devueltos de este verbo:

- El ID de sesión, devuelto a la aplicación en el parámetro *lua_sid* (y como valor de retorno desde el punto de entrada SLI)
- El nombre de LU, devuelto a la aplicación en el parámetro *lua_luname*

Uso y restricciones

Una vez que el verbo `SLI_OPEN` ha finalizado correctamente, esta sesión utiliza la LU para la que se ha iniciado la sesión. Ninguna otra sesión LUA (de esta u otra aplicación) puede utilizar la LU hasta que se emite el verbo `SLI_CLOSE` o hasta que se recibe el código de retorno primario `LUA_SESSION_FAILURE`.

Si el verbo `SLI_OPEN` devuelve un código de retorno primario `LUA_IN_PROGRESS`, el ID de sesión se devolverá en el parámetro *lua_sid*. Este ID de sesión es el mismo que el que se devuelve cuando el verbo finaliza correctamente y se puede utilizar para emitir otros verbos en la sesión.

SLI_PURGE

El verbo `SLI_PURGE` cancela un `SLI_RECEIVE` anterior. Un `SLI_RECEIVE` puede esperar indefinidamente si se envía sin utilizar la opción *lua_flag1.nowait* (retorno inmediato) y no hay datos disponibles en el flujo especificado; `SLI_PURGE` fuerza el retorno del verbo en espera (con el código de retorno primario `LUA_CANCELLED`).

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

`LUA_VERB_SLI`

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

`LUA_OPCODE_SLI_PURGE`

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa, según lo devuelto en el verbo `SLI_OPEN`.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un ID devuelto en un verbo SLI_OPEN anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_data_ptr

Puntero al VCB SLI_RECEIVE que se va a eliminar.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.



Para ver más información, consulte el Capítulo 2, “Diseño y desarrollo de aplicaciones LUA”, en la página 13.

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, se devuelven los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

SLI_PURGE

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

Se ha emitido un verbo SLI_CLOSE mientras este verbo estaba pendiente.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* tiene el valor 0 (cero).

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc

Los valores posibles son:

LUA_NO_RECEIVE_TO_PURGE

El parámetro *lua_data_ptr* no se ha establecido en la dirección de un VCB de SLI_RECEIVE anterior.

LUA_NO_SLI_SESSION

Un verbo SLI_OPEN todavía no ha finalizado correctamente para la LU especificada en este verbo o ha fallado la sesión.

LUA_SLI_PURGE_PENDING

Ya había un SLI_PURGE pendiente cuando se ha emitido este verbo. Tan solo puede haber un SLI_PURGE pendiente a la vez.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc

LUA_UNSUCCESSFUL

lua_sec_rc

Los valores posibles son:

LUA_INVALID_PROCESS

El proceso del sistema operativo que ha emitido este verbo no es el mismo que el proceso que ha emitido el verbo SLI_OPEN para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_NO_RECEIVE_TO_PURGE

El verbo SLI_RECEIVE anterior ha finalizado antes de que la aplicación emitiera SLI_PURGE. No se trata de una condición de error, por lo que el programa de aplicación debería diseñarse para manejar esta acción sin informar de errores.

LUA_NAU_INOPERATIVE

Un componente SNA necesario (como la LU de LUA) no está activo o tiene un estado anormal.

LUA_NO_SESSION

La sesión SNA con la LU remota no está activa.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

lua_prim_rc

SLI_PURGE

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RECEIVED_UNBIND

Este código de retorno indica que el sistema principal ha enviado un mandato UNBIND para finalizar la sesión. Este valor sólo se puede producir si el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

Este verbo sólo puede utilizarse cuando se ha emitido un verbo SLI_RECEIVE y todavía está pendiente de finalización (es decir, el código de retorno primario es IN_PROGRESS).

SLI_RECEIVE

El verbo SLI_RECEIVE recibe una cadena completa de datos o información de estado, enviada desde el sistema principal a la LU de la aplicación.

Puede especificar un determinado flujo de mensajes (LU normal, LU urgente, SSCP normal o SSCP urgente) del que se leerán los datos, o bien especificar más de un flujo de mensajes. Puede tener varios verbos SLI_RECEIVE pendientes, siempre y cuando no haya dos que especifiquen el mismo flujo.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.*lua_opcode*

LUA_OPCODE_SLI_RECEIVE

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa, según lo devuelto en el verbo SLI_OPEN.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un ID devuelto en un verbo SLI_OPEN anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_max_length

Longitud del almacenamiento intermedio suministrado para recibir los datos.

lua_data_ptr

Puntero al almacenamiento intermedio suministrado para recibir los datos.

lua_post_handle

| |
|------------|
| AIX, LINUX |
|------------|

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

| |
|---------|
| WINDOWS |
|---------|

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.

| |
|--|
| |
|--|

Para ver más información, consulte el Capítulo 2, "Diseño y desarrollo de aplicaciones LUA", en la página 13.

Parámetros *lua_flag1*

Establezca el parámetro *lua_flag1.nowait* en 1 si desea que el verbo SLI_RECEIVE vuelva tan pronto como sea posible, tanto si hay datos disponibles para leer como si no, o establézcalo en 0 (cero) si desea que el verbo espere hasta que reciba datos antes de volver.

Nota:

1. El hecho de establecer el parámetro *lua_flag1.nowait* en 1 no significa que el verbo finalizará de forma síncrona. La biblioteca LUA necesita comunicarse con el nodo local para determinar si hay datos disponibles y para ello se requiere un retorno asíncrono de verbo para evitar el bloqueo de la aplicación. El parámetro significa que, si no hay datos disponibles de inmediato, el verbo asíncrono devolverá tan pronto como sea posible para indicarlo.
2. Si la primera RU de una cadena de varias RU está disponible cuando la aplicación emite SLI_RECEIVE, el parámetro *lua_flag1.nowait* se ignora; SLI_RECEIVE espera a que llegue la cadena completa de datos antes de volver.

Establezca el parámetro *lua_flag1.bid_enable* en 1 para volver a activar el verbo SLI_BID más reciente (lo que equivale a volver a emitir SLI_BID con exactamente los mismos parámetros que antes), o establézcalo en 0 (cero) si no desea volver a activar SLI_BID. Si se vuelve a activar el verbo SLI_BID anterior, se vuelve a utilizar el VCB originalmente asignado para éste, de modo que este VCB no debe haberse liberado ni modificado. (Para ver más información, consulte “Interacción con otros verbos” en la página 143.)

Establezca uno o varios de los siguientes indicadores a 1 para indicar de qué flujo de mensajes se deben leer los datos:

lua_flag1.sscp_exp

lua_flag1.lu_exp

lua_flag1.sscp_norm

lua_flag1.lu_norm

Si se establece más de un indicador, se devolverán los datos disponibles cuya prioridad sea más alta. El orden de prioridad (de más alta a más baja) es: SSCP urgente, LU urgente, SSCP normal, LU normal. Se establecerá el indicador equivalente en el grupo *lua_flag2* para indicar de qué flujo se han leído los datos (consulte “Parámetros devueltos”).

La implementación de Communications Server para Linux de LUA no devuelve datos en el flujo acelerado de SSCP. La aplicación puede establecer el indicador *sscp_exp* para garantizar la compatibilidad con otras implementaciones de LUA, pero nunca se devolverán datos en este flujo.

Parámetros devueltos

LUA siempre devuelve los siguientes parámetros:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

lua_flag2.bid_enable

Este parámetro tiene el valor 1 si se ha vuelto a activar correctamente un verbo SLI_BID o el valor 0 si no se ha vuelto a activar.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta o datos truncados

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc
LUA_OK

Si el verbo finaliza de forma correcta, se devuelven los parámetros siguientes. También se devuelven si el verbo vuelve con datos truncados porque el parámetro *lua_data_length* suministrado era demasiado pequeño (consulte “Otras condiciones” en la página 141).

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_data_length

Longitud de los datos recibidos. LUA coloca los datos en el almacenamiento intermedio especificado por *lua_data_ptr*.

Si *lua_rh.rrr* está desactivado (unidad de petición) y *lua_rh.sdi* está activado (datos de detección incluidos), esto indica que LUA ha convertido una unidad de petición enviada por el sistema principal en una petición de excepción (EXR). En este caso, los bytes 0–3 del almacenamiento intermedio de datos contienen los datos de detección asociados con la excepción y los bytes 4–6 contienen hasta los 3 primeros bytes de la unidad de petición original.

lua_th Información de la cabecera de transmisión (TH) del mensaje recibido.

lua_rh Información de la cabecera de petición/respuesta (RH) del mensaje recibido.

lua_message_type

Tipo de mensaje recibido, que puede ser uno de los siguientes:

LUA_MESSAGE_TYPE_LU_DATA
LUA_MESSAGE_TYPE_SSCP_DATA
LUA_MESSAGE_TYPE_RSP
LUA_MESSAGE_TYPE_BID
LUA_MESSAGE_TYPE_BIS
LUA_MESSAGE_TYPE_CANCEL
LUA_MESSAGE_TYPE_CHASE
LUA_MESSAGE_TYPE_LUSTAT_LU
LUA_MESSAGE_TYPE_LUSTAT_SSCP
LUA_MESSAGE_TYPE_QC
LUA_MESSAGE_TYPE_QEC
LUA_MESSAGE_TYPE_RELQ
LUA_MESSAGE_TYPE_RTR
LUA_MESSAGE_TYPE_SBI
LUA_MESSAGE_TYPE_SIGNAL

Parámetros de *lua_flag2*

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han recibido los datos:

lua_flag2.lu_exp

lua_flag2.sscp_norm

lua_flag2.lu_norm

La implementación de Communications Server para Linux de LUA no devuelve datos en el flujo rápido de SSCP y, por lo tanto, el indicador *sscp_exp* no se establecerá nunca (aunque lo puedan establecer otras implementaciones de LUA).

Ejecución correcta: información de estado

Nota: SLI_RECEIVE sólo puede devolver información de estado si no hay ningún verbo SLI_BID pendiente. Si ambos verbos están en proceso cuando la información de estado está disponible, el estado se devuelve para el verbo SLI_BID y SLI_RECEIVE permanece en proceso.

Si el verbo devuelve información de estado de LUA en lugar de datos, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_STATUS

lua_sec_rc

LUA_READY

La sesión SLI está preparada para procesar mandatos adicionales. Este estado se utiliza después de haber informado de un estado LUA_NOT_READY anterior o después de haber finalizado un verbo SLI_CLOSE con *lua_prim_rc* establecido en LUA_CANCELLED y *lua_sec_rc* establecido en RECEIVED_UNBIND_HOLD o RECEIVED_UNBIND_NORMAL.

LUA_NOT_READY

La sesión SLI se ha suspendido temporalmente por uno de los motivos siguientes:

- Se ha recibido un mandato CLEAR. La sesión se reanuda cuando se recibe un mandato SDT.
- Se ha recibido un mandato UNBIND de tipo X'02' (BIND próximo). La sesión se suspende hasta que se reciben los mandatos BIND, CRV y STSN opcionales y SDT; se reanuda después del SDT. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo SLI_OPEN original.
- Se ha recibido un mandato UNBIND de tipo X'01' (normal) y el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED. La sesión se suspende hasta que se reciben los mandatos BIND, CRV y STSN opcionales y SDT; se reanuda después del SDT. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo SLI_OPEN original.

La aplicación debería emitir otro SLI_BID o SLI_RECEIVE para recibir el estado READY cuando la sesión se reanuda. Puede seguir

emitiendo verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP aunque el estado de la sesión sea LUA_NOT_READY.

LUA_INIT_COMPLETE

La aplicación ha emitido SLI_OPEN con el tipo LUA_OPEN_TYPE_PRIM_SSCP y el verbo RUI_INIT subyacente ha finalizado ahora. La aplicación ahora puede emitir verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP.

LUA_SESSION_END_REQUESTED

El sistema principal ha enviado un mandato SHUTD para solicitar a la aplicación que cierre la sesión. La aplicación debería emitir SLI_CLOSE tan pronto como esté preparada para cerrar la sesión.

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo o un mensaje desde el sistema principal lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc
Los valores posibles son:

LUA_PURGED

Este verbo SLI_RECEIVE se ha cancelado mediante un verbo SLI_PURGE.

LUA_TERMINATED

Se ha emitido un verbo SLI_CLOSE mientras este verbo estaba pendiente.

LUA_CANCEL_COMMAND_RECEIVED

El sistema principal ha enviado un mandato CANCEL para cancelar el resto de la cadena de datos que se están recibiendo.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc
Los valores posibles son:

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* contiene un valor que no es válido.

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_BID_ALREADY_ENABLED

Se ha establecido el parámetro *lua_flag1.bid_enable* para volver a activar un verbo SLI_BID, pero el verbo SLI_BID anterior todavía se estaba procesando.

LUA_INVALID_FLOW

No se ha establecido ninguno de los indicadores de flujo *lua_flag1*. Como mínimo uno de estos indicadores debe tener el valor 1 para indicar de qué flujo o flujos se debe leer.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_NO_PREVIOUS_BID_ENABLED

Se ha establecido el parámetro *lua_flag1.bid_enable* para volver a habilitar un verbo SLI_BID, pero no se ha podido habilitar ningún verbo SLI_BID anterior. (Para ver más información, consulte “Interacción con otros verbos” en la página 143.)

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
Los valores posibles son:

LUA_NO_SLI_SESSION

Un verbo SLI_OPEN todavía no ha finalizado correctamente para la LU especificada en este verbo o ha fallado la sesión.

LUA_RECEIVE_ON_FLOW_PENDING

Los dos indicadores del grupo *lua_flag1* especifican uno o varios flujos de sesión para los que ya había un verbo SLI_RECEIVE pendiente. No puede haber más de un verbo SLI_RECEIVE pendiente simultáneamente en un flujo de sesión.

Respuesta negativa enviada al sistema principal: El siguiente código de retorno primario indica que Communications Server para Linux ha detectado un error en los datos recibidos del sistema principal. En lugar de pasar el mensaje recibido a la aplicación en un verbo SLI_RECEIVE, Communications Server para Linux elimina el mensaje y envía una respuesta negativa al sistema principal. LUA informa a la aplicación en un verbo SLI_RECEIVE o SLI_BID posterior de que se ha enviado una respuesta negativa.

lua_prim_rc
LUA_NEGATIVE_RSP

lua_sec_rc
El código de detección enviado al sistema principal en la respuesta negativa. Esto indica que Communications Server para Linux ha detectado un error en los datos de sistema principal y ha enviado una respuesta negativa al sistema principal. Para ver información sobre cómo interpretar los valores de código de detección que pueden devolverse, consulte "Información de SNA" en la página 36.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
Los valores posibles son:

LUA_DATA_TRUNCATED

El parámetro *lua_data_length* es menor que la longitud real de los datos recibidos en el mensaje. Sólo se han devuelto *lua_data_length* bytes de datos al verbo; el resto de datos se ha eliminado. También se devuelven parámetros adicionales si se obtiene este código de retorno secundario; consulte "Ejecución correcta o datos truncados" en la página 137.

LUA_NO_DATA

El parámetro *lua_flag1.nowait* indica el retorno inmediato sin esperar a los datos, y no hay datos disponibles en el flujo o flujos de sesión especificados.

LUA_INVALID_PROCESS

El proceso del sistema operativo que ha emitido este verbo no es el mismo que el proceso que ha emitido el verbo SLI_OPEN para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_NAU_INOPERATIVE

Un componente SNA necesario (como la LU de LUA) no está activo o tiene un estado anormal.

LUA_NO_SESSION

La sesión SNA con la LU remota no está activa.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

SLI_RECEIVE

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RECEIVED_UNBIND

Este código de retorno indica que el sistema principal ha enviado un mandato UNBIND para finalizar la sesión. Este valor sólo se puede producir si el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED.

LUA_RUI_WRITE_FAILURE

Un verbo RUI_WRITE utilizado para procesar este verbo SLI ha fallado con un código de retorno de error inesperado.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

El verbo SLI_OPEN debe completarse correctamente para que se pueda emitir este verbo.

Si hay un SLI_RECEIVE pendiente, puede emitir otro SLI_RECEIVE sólo si éste especifica un flujo o varios flujos de sesión diferentes de los SLI_RECEIVE pendientes; no puede haber más de un SLI_RECEIVE pendiente para el mismo flujo de sesión.

El parámetro *lua_flag1.bid_enable* sólo puede utilizarse si se cumple lo siguiente:

- SLI_BID se ha emitido correctamente y ha finalizado
- El almacenamiento asignado para el verbo SLI_BID no se ha liberado ni modificado
- No hay ningún otro verbo SLI_BID pendiente

Si utiliza este parámetro para volver a habilitar un verbo SLI_BID anterior, debe seguir habiendo como mínimo un indicador de flujo de mensajes en SLI_RECEIVE, a fin de indicar el flujo o flujos en que la aplicación aceptará datos. Si los primeros datos que se van a recibir están en un flujo aceptado por el verbo SLI_RECEIVE, se devolverá SLI_RECEIVE con estos datos, pero no habrá retorno de SLI_BID. De lo contrario, se devolverá SLI_BID para indicar que hay datos que se deben leer (como SLI_BID acepta datos en todos los flujos, siempre aceptará los datos si SLI_RECEIVE no lo hace). En este caso, la aplicación debe emitir otro SLI_RECEIVE en el flujo apropiado para obtener los datos.

Si desea utilizar SLI_BID para gestionar datos en todos los flujos, en lugar de tener los datos en un determinado flujo gestionado preferentemente por SLI_RECEIVE en lugar de por SLI_BID, deberá volver a emitir SLI_BID explícitamente en vez de utilizar SLI_RECEIVE para volver a activar el verbo SLI_BID anterior.

Uso y restricciones

Si los datos recibidos superan la longitud máxima especificada en el parámetro *lua_max_length*, se truncarán; sólo se devolverán *lua_max_length* bytes de datos. También se devolverán los códigos de retorno primario y secundario LUA_UNSUCCESSFUL y LUA_DATA_TRUNCATED.

Si el verbo SLI_RECEIVE establece bits en *lua_flag1* para aceptar datos en más de un flujo y hay datos disponibles en más de uno de los flujos especificados, los datos del flujo con la prioridad más alta se devolverán a la aplicación. Las prioridades de flujo se indican a continuación (de la más alta a la más baja):

- SSCP rápido
- LU rápido
- SSCP normal
- LU normal

Una vez que se ha leído un mensaje utilizando el verbo SLI_RECEIVE, éste se elimina de la cola de mensajes de entrada y no puede volver a accederse a dicho mensaje. La aplicación puede utilizar SLI_BID como una lectura no destructiva para comprobar el tipo de datos disponibles y determinar cómo procesarlos, y después emitir un SLI_RECEIVE posterior para recopilar los datos. Sin embargo, si emite SLI_RECEIVE con varios indicadores *lua_flag1* establecidos para aceptar datos en más de un flujo, puede recibir un mensaje distinto del mensaje identificado en SLI_BID, si han llegado datos en un flujo de prioridad más alta

SLI_RECEIVE

entre los verbos SLI_BID y SLI_RECEIVE. Para asegurarse de que recibe el mismo mensaje identificado en SLI_BID, debería establecer los identificadores *lua_flag1* en SLI_RECEIVE para aceptar datos únicamente en el flujo identificado en la respuesta de SLI_BID.

El ritmo puede utilizarse en la semisesión primaria a secundaria (esto se especifica en la configuración del sistema principal) a fin de evitar que la aplicación LUA se vea desbordada por un número excesivo de mensajes. Si la aplicación LUA es lenta al leer mensajes, Communications Server para Linux retarda el envío de las respuestas de ritmo al sistema principal a fin de reducir la velocidad del mismo.

SLI_SEND

El verbo SLI_SEND envía una unidad de petición o respuesta SNA desde la aplicación LUA al sistema principal a través de la sesión de LU o de la sesión de SSCP.

Una aplicación puede tener como máximo dos verbos SLI_SEND pendientes a la vez, los cuales deben estar en flujos de sesión diferentes.

Parámetros proporcionados

La aplicación proporciona los siguientes parámetros:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

Debe tener el valor `sizeof(LUA_VERB_RECORD)`.

lua_opcode

LUA_OPCODE_SLI_SEND

lua_correlator

Opcional. Valor de 4 bytes que puede utilizar para correlacionar este verbo con otro proceso de la aplicación. LUA no utiliza ni cambia esta información.

lua_luname

Nombre en ASCII de la LU utilizada por la sesión. Debe coincidir con el nombre de LU de una sesión LUA activa, según lo devuelto en el verbo SLI_OPEN.

Este parámetro es necesario sólo si el parámetro *lua_sid* es 0 (cero). Si se proporciona un identificador de sesión en *lua_sid*, LUA no utiliza este parámetro.

Este parámetro debe tener ocho bytes de longitud; rellénelo por la derecha con espacios en blanco, 0x20, si el nombre tiene menos de ocho caracteres.

lua_sid Identificador de la sesión. Debe coincidir con un ID devuelto en un verbo SLI_OPEN anterior.

Este parámetro es opcional; si no especifica el identificador de sesión, debe especificar el nombre de LU para la sesión en el parámetro *lua_luname*.

lua_data_length

Longitud de los datos proporcionados.

Cuando se envía una respuesta afirmativa, este parámetro normalmente tiene el valor 0 (cero). LUA finalizará la respuesta en función del número de secuencia suministrado. En el caso de una respuesta afirmativa a un BIND o STSN, se permite una respuesta ampliada, de modo que puede utilizarse un valor distinto de cero.

Cuando se envía una respuesta negativa, establezca este parámetro a la longitud del código de detección SNA (cuatro bytes), que se suministra en el almacenamiento intermedio de datos.

lua_data_ptr

Puntero al almacenamiento intermedio que contiene los datos suministrados.

Para una petición o una respuesta afirmativa que requiere datos, el almacenamiento intermedio debe contener toda la RU. La longitud de la RU debe especificarse en *lua_data_length*.

Para una respuesta negativa, el almacenamiento intermedio debe contener el código de detección SNA.

lua_post_handle

AIX, LINUX

Puntero a una rutina de devolución de llamada que LUA llamará para indicar la finalización si el verbo se completa de forma asíncrona.

WINDOWS

Si el VCB se utiliza en una llamada de función SLI, establezca este campo en un manejador de sucesos. Si el VCB se utiliza en una llamada de función WinSLI, este campo está reservado.

████████

Para ver más información, consulte el Capítulo 2, "Diseño y desarrollo de aplicaciones LUA", en la página 13.

lua_th.snf

Necesario sólo cuando se envía una respuesta. El número de secuencia de la petición cuya respuesta es ésta.

lua_rh Cuando se envía una petición, la mayoría de los bits *lua_rh* deben establecerse para corresponder con la cabecera de petición (RH) del mensaje que se debe enviar. No establezca *lua_rh.pi* y *lua_rh.qri*; lo hará LUA.

Cuando se envía una respuesta, sólo se utilizan los dos bits *lua_rh* siguientes. Los demás deben tener el valor 0 (cero). Los bits *lua_rh* son:

lua_rh.rrl

Establézcalo a 1 para indicar una respuesta

lua_rh.ri

Establézcalo a 0 para una respuesta afirmativa, o a 1 para una respuesta negativa

Parámetros *lua_flag1*

Establezca uno de los siguientes indicadores a 1 para indicar en qué flujo de mensajes se enviarán los datos:

lua_flag1.lu_exp

lua_flag1.sscp_norm

lua_flag1.lu_norm

Únicamente uno de los indicadores se debe establecer en 1. Communications Server para Linux no permite a las aplicaciones enviar datos en el flujo rápido de SSCP (el indicador *lua_flag1.sscp_exp*).

lua_message_type

Tipo de mensaje del mensaje que se va a enviar. Los valores posibles son:

LUA_MESSAGE_TYPE_LU_DATA

LUA_MESSAGE_TYPE_SSCP_DATA

LUA_MESSAGE_TYPE_RSP

LUA_MESSAGE_TYPE_BID

LUA_MESSAGE_TYPE_BIS

LUA_MESSAGE_TYPE_CANCEL

LUA_MESSAGE_TYPE_CHASE

LUA_MESSAGE_TYPE_LUSTAT_LU

LUA_MESSAGE_TYPE_LUSTAT_SSCP

LUA_MESSAGE_TYPE_QC

LUA_MESSAGE_TYPE_QEC

LUA_MESSAGE_TYPE_RELQ

LUA_MESSAGE_TYPE_RTR

LUA_MESSAGE_TYPE_SBI

Parámetros devueltos

LUA siempre devuelve el siguiente parámetro:

lua_flag2.async

Este indicador tiene el valor 1 si el verbo ha finalizado de manera asíncrona, o 0 (cero) si ha finalizado de manera síncrona.

Los otros parámetros devueltos dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta

Si el verbo finaliza de forma correcta, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_sid Si la aplicación ha especificado el parámetro *lua_luname* al emitir este verbo, en lugar de especificar el ID de sesión, LUA proporciona el ID de sesión.

lua_th TH finalizada del mensaje escrito, incluidos los campos rellenos por

LUA. Quizás deba guardar el valor de *lua_th.snf* (el número de secuencia) para la correlación con las respuestas del sistema principal.

Parámetros de *lua_flag2*

Uno de los siguientes indicadores tendrá el valor 1 para indicar en qué flujo de mensajes se han enviado los datos:

lua_flag2.lu_exp

lua_flag2.sscp_norm

lua_flag2.lu_norm

La implementación de Communications Server para Linux de LUA no permite a las aplicaciones enviar datos en el flujo rápido de SSCP y, por consiguiente, nunca establecerá el indicador *sscp_exp* (aunque es posible que lo establezcan otras implementaciones de LUA).

lua_sequence_number

Número de secuencia de la RU que LUA utiliza para enviar los datos (o de la primera RU, si los datos necesitan una cadena de RU). Se guarda en formato de línea.

Ejecución correcta: información de estado

Si el verbo devuelve información de estado de LUA, LUA devuelve los parámetros siguientes:

lua_prim_rc

LUA_STATUS

lua_sec_rc

LUA_READY

La sesión SLI está preparada para procesar mandatos adicionales. Este estado se utiliza después de haber informado de un estado `LUA_NOT_READY` anterior o después de haber finalizado un verbo `SLI_CLOSE` con *lua_prim_rc* establecido en `LUA_CANCELLED` y *lua_sec_rc* establecido en `RECEIVED_UNBIND_HOLD` o `RECEIVED_UNBIND_NORMAL`.

LUA_NOT_READY

La sesión SLI se ha suspendido temporalmente por uno de los motivos siguientes:

- Se ha recibido un mandato `CLEAR`. La sesión se reanuda cuando se recibe un mandato `SDT`.
- Se ha recibido un mandato `UNBIND` de tipo `X'02'` (`BIND` próximo). La sesión se suspende hasta que se reciben los mandatos `BIND`, `CRV` y `STSN` opcionales y `SDT`; se reanuda después del `SDT`. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo `SLI_OPEN` original.
- Se ha recibido un mandato `UNBIND` de tipo `X'01'` (normal) y el verbo `SLI_OPEN` para esta sesión especificaba *lua_session_type* `LUA_SESSION_TYPE_DEDICATED`. La sesión se suspende hasta que se reciben los mandatos `BIND`, `CRV` y `STSN` opcionales y `SDT`; se reanuda después del `SDT`. Se volverá a llamar a cualquier rutina de ampliación del usuario proporcionada por el verbo `SLI_OPEN` original.

La aplicación debería emitir `SLI_BID` o `SLI_RECEIVE` para recibir el estado `READY` cuando la sesión se reanuda. Puede seguir

emitiendo verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP aunque el estado de la sesión sea LUA_NOT_READY.

LUA_INIT_COMPLETE

La aplicación ha emitido SLI_OPEN con el tipo LUA_OPEN_TYPE_PRIM_SSCP y el verbo RUI_INIT subyacente ha finalizado ahora. La aplicación ahora puede emitir verbos SLI_SEND y SLI_RECEIVE para datos de flujo normal de SSCP.

LUA_SESSION_END_REQUESTED

El sistema principal ha enviado un mandato SHUTD para solicitar a la aplicación que cierre la sesión. La aplicación debería emitir SLI_CLOSE tan pronto como esté preparada para cerrar la sesión.

Ejecución incorrecta

Si un verbo no finaliza de forma correcta, LUA devuelve un código de retorno primario para indicar el tipo de error y un código de retorno secundario para proporcionar datos específicos sobre por qué la ejecución no ha sido correcta.

Verbo cancelado: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque otro verbo lo ha cancelado:

lua_prim_rc
LUA_CANCELLED

lua_sec_rc

LUA_TERMINATED

El verbo se ha cancelado porque se ha emitido un verbo SLI_CLOSE para esta sesión.

Comprobación de parámetros: Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente porque hay un parámetro incorrecto:

lua_prim_rc
LUA_PARAMETER_CHECK

lua_sec_rc

Los valores posibles son:

LUA_BAD_DATA_PTR

El parámetro *lua_data_ptr* contiene un valor que no es válido.

LUA_BAD_SESSION_ID

El parámetro *lua_sid* no coincide con el identificador de sesión de ninguna sesión de LU LUA activa.

LUA_INVALID_FLOW

Más de uno de los indicadores de flujo *lua_flag1* tienen el valor 1. Únicamente uno de estos indicadores debe tener el valor 1 para indicar en qué flujo de sesión se van a enviar los datos.

Se ha establecido el indicador de flujo *lua_flag1.sscp_exp*, que indica que el mensaje debe enviarse en el flujo rápido de SSCP. Communications Server para Linux no permite que las aplicaciones envíen datos en este flujo.

LUA_INVALID_MESSAGE_TYPE

El parámetro *lua_message_type* no se ha establecido en un valor válido.

AIX, LINUX

LUA_INVALID_POST_HANDLE

El parámetro *lua_post_handle* no es un puntero válido a una rutina de devolución de llamada.

LUA_REQUIRED_FIELD_MISSING

No se ha establecido ninguno de los indicadores de flujo *lua_flag1*. Únicamente uno de estos indicadores debe tener el valor 1.

LUA_RESERVED_FIELD_NOT_ZERO

Un campo reservado en el registro de verbo, o un parámetro no utilizado por este verbo, tiene un valor distinto de cero.

LUA_VERB_LENGTH_INVALID

El valor del parámetro *lua_verb_length* es inferior a la longitud del registro de verbo necesaria para este verbo.

LUA_DATA_LENGTH_ERROR

La aplicación ha utilizado SLI_SEND para enviar LUSTAT al sistema principal, pero no ha proporcionado los 4 bytes necesarios de información de estado.

Comprobación de estado: Los siguientes códigos de retorno indican que el verbo se ha emitido en un estado de sesión en que no es válido:

lua_prim_rc
LUA_STATE_CHECK

lua_sec_rc
Los valores posibles son:

LUA_MAX_NUMBER_OF_SENDS

La aplicación ya tenía dos verbos SLI_SEND en proceso cuando ha emitido este verbo. Una aplicación puede tener como máximo dos verbos SLI_SEND pendientes a la vez, los cuales deben estar en flujos de sesión diferentes.

LUA_NO_SLI_SESSION

Un verbo SLI_OPEN todavía no ha finalizado correctamente para la LU especificada en este verbo o ha fallado la sesión.

LUA_SEND_ON_FLOW_PENDING

Un verbo SLI_SEND ya estaba pendiente para el flujo de sesión especificado en este verbo (el flujo de sesión se especifica estableciendo uno de los indicadores de flujo *lua_flag1* en 1). No puede haber más de un verbo SLI_SEND pendiente simultáneamente en un flujo de sesión.

Otras condiciones: Los siguientes códigos de retorno indican que el registro de verbo era válido, pero que el verbo no se ha completado correctamente:

lua_prim_rc
LUA_UNSUCCESSFUL

lua_sec_rc
Los valores posibles son:

LUA_INVALID_PROCESS

El proceso del sistema operativo que ha emitido este verbo no es el mismo que el proceso que ha emitido el verbo SLI_OPEN para esta sesión. Sólo el proceso que ha iniciado una sesión puede emitir verbos en esa sesión.

LUA_INVALID_SESSION_PARAMETERS

La aplicación ha utilizado SLI_SEND para enviar una respuesta afirmativa a un mensaje BIND recibido del sistema principal. Sin embargo, el nodo de Communications Server para Linux no puede aceptar los parámetros BIND como se han especificado y ha enviado una respuesta negativa al sistema principal. Para obtener más información sobre los perfiles de BIND aceptados por Communications Server para Linux, consulte el apartado "Información de SNA" en la página 36.

LUA_RSP_CORRELATION_ERROR

Se ha utilizado SLI_SEND para enviar una respuesta, pero el parámetro *lua_th.snf* (que indica el número de secuencia del mensaje recibido al que se responde) no contiene un valor válido.

LUA_RU_LENGTH_ERROR

El parámetro *lua_data_length* contiene un valor que no es válido. Cuando se envían datos en el flujo LU normal, la longitud máxima es la especificada en el BIND recibido del sistema principal; para todos los demás flujos la longitud máxima es de 256 bytes.

LUA_NAU_INOPERATIVE

Un componente SNA necesario (como la LU de LUA) no está activo o tiene un estado anormal.

LUA_NO_SESSION

La sesión SNA con la LU remota no está activa.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

(cualquier otro valor)

Cualquier otro código de retorno secundario que aparezca aquí es un código de detección SNA que indica que los datos SNA suministrados no son válidos o no se han podido enviar. Para ver información sobre cómo interpretar los códigos de detección SNA que pueden devolverse, consulte "Información de SNA" en la página 36.

Los siguientes códigos de retorno indican que el verbo no ha finalizado correctamente por otros motivos:

lua_prim_rc

LUA_COMM_SUBSYSTEM_ABENDED

Un componente de software Communications Server para Linux

necesario (por ejemplo el nodo) ha terminado o se ha detenido. Póngase en contacto con el administrador del sistema si es necesario.

lua_prim_rc

LUA_COMM_SUBSYSTEM_NOT_LOADED

El software Remote API Client no se ha iniciado o el nodo no se ha iniciado o configurado correctamente para las aplicaciones LUA. Compruebe los parámetros de configuración de LUA de Communications Server para Linux e inicie Remote API Client y el nodo antes de ejecutar la aplicación.

lua_prim_rc

LUA_SESSION_FAILURE

La sesión LUA ha fallado. Para reiniciarla, la aplicación puede volver a emitir SLI_OPEN.

lua_sec_rc

Los valores posibles son:

LUA_LU_COMPONENT_DISCONNECTED

Este código de retorno indica que la sesión LUA ha fallado debido a un problema con el enlace de comunicaciones o con la LU del sistema principal.

LUA_RECEIVED_UNBIND

Este código de retorno indica que el sistema principal ha enviado un mandato UNBIND para finalizar la sesión. Este valor sólo se puede producir si el verbo SLI_OPEN para esta sesión especificaba *lua_session_type* LUA_SESSION_TYPE_DEDICATED.

LUA_SLI_LOGIC_ERROR

Este código de retorno indica una de las siguientes condiciones:

- El sistema principal ha violado los protocolos SNA
- Se ha detectado un error interno en LUA

Intente reproducir el problema con el rastreo SNA activo (póngase en contacto con el administrador del sistema si es necesario) y compruebe que el sistema principal está enviando los datos correctos. Si esto no soluciona el problema, póngase en contacto con el personal de soporte de Communications Server para Linux.

lua_prim_rc

LUA_INVALID_VERB

El parámetro *lua_verb* o *lua_opcode* no es válido. El verbo no se ha ejecutado.

lua_prim_rc

LUA_STACK_TOO_SMALL

El tamaño de pila de la aplicación es demasiado pequeño para que LUA finalice la petición. Aumente el tamaño de pila de la aplicación.

lua_prim_rc

LUA_UNEXPECTED_DOS_ERROR

Se ha producido un error del sistema operativo.

lua_sec_rc

Este valor es el código de retorno del sistema operativo. Compruebe la documentación del sistema operativo para conocer el significado de este código de retorno.

Interacción con otros verbos

El verbo SLI_OPEN debe haberse emitido correctamente para que se pueda emitir este verbo.

Mientras hay un verbo SLI_SEND pendiente, puede emitir un segundo verbo SLI_SEND sólo si éste especifica un flujo de sesión diferente al del SLI_SEND pendiente; es decir, no puede haber más de un SLI_SEND pendiente para el mismo flujo de sesión. No puede haber más de dos SLI_SEND pendientes en total.

El verbo SLI_SEND se puede emitir en el flujo normal de SSCP en cualquier momento después de un verbo SLI_OPEN satisfactorio que especifica inicio primario de sesión con acceso a SSCP. Los verbos SLI_SEND en otros flujos o para otros tipos de inicio de sesión sólo están permitidos después de haberse recibido un BIND y deben estar soportados por los protocolos especificados en el BIND.

Uso y restricciones

La Tabla 2 muestra los valores válidos para distintos parámetros en SLI_SEND, según el tipo de mensaje SNA que se envíe.

Tabla 2. Valores de parámetros de SLI_SEND según el tipo de mensaje

| Parámetro de SLI_SEND | LU_DATA, SSCP_DATA | RSP | BID, BIS, RTR | CHASE QC | QEC, RELQ, SBL, SIG | RQR | LUSTAT_LU, LUSTAT_SSCP |
|--|---|------------------------------------|---------------|--------------------|---------------------|------|---|
| <i>lua_rh</i> | FI, DR1I, DR2I, RI, BBI, EBI, CDI, CSI, EDI | RI | SDI, QRI | SDI, QRI, EBI, CDI | SDI | 0 | SDI, QRI, DR1I, DR2I, RI, BBI, EBI, CDI |
| <i>lua_th</i> | 0 | SNF | 0 | 0 | 0 | 0 | 0 |
| <i>lua_data_ptr</i> | Obligatorio (nulo si no hay datos) | Obligatorio (nulo si no hay datos) | nulo | nulo | nulo | nulo | Obligatorio |
| <i>lua_data_length</i> | Obligatorio | Obligatorio (0 si no hay datos) | 0 | 0 | 0 | 0 | Obligatorio |
| Indicadores de flujo <i>lua_flag1</i> | 0 | Obligatorio (establecer uno) | 0 | 0 | 0 | 0 | 0 |

Cuando la aplicación envía una respuesta SNA, debe hacer lo que se indica a continuación. LUA encontrará el código de petición adecuado basándose en el número de secuencia proporcionado.

- Establezca *lua_message_type* en LUA_MESSAGE_TYPE_RSP.
- Establezca *lua_th.snf* en el número de secuencia de la petición cuya respuesta es ésta.
- Establezca el indicador de flujo *lua_flag1* adecuado.
- Para una respuesta afirmativa que únicamente requiera código de petición, establezca *lua_rh.ri* y *lua_data_length* en 0 (cero).
- Para una respuesta negativa:
 - Establezca *lua_rh.ri* en 1.
 - Establezca *lua_data_ptr* para apuntar a un código de detección SNA adecuado.
 - Establezca *lua_data_length* en 4 (la longitud del código de detección).

La finalización correcta de SLI_SEND indica que el mensaje se ha puesto en cola correctamente para el enlace de datos; esto no indica necesariamente que el mensaje se haya enviado correctamente o que el sistema principal lo haya aceptado.

Se puede utilizar ritmo en la semisesión secundaria a primaria (esto se especifica en BIND), a fin de evitar que la aplicación LUA envíe más datos de los que puede manejar la LU de Communications Server para Linux o la LU de sistema principal. Si se da este caso, puede que LUA retrase un verbo SLI_SEND en el flujo normal de LU y que tarde un poco en completarse.

SLI_BIND_ROUTINE

Este verbo se envía desde LUA a la aplicación (utilizando el punto de entrada de la rutina de ampliación BIND proporcionado por la aplicación en el verbo SLI_OPEN) y no desde la aplicación a LUA.

El verbo SLI_BIND_ROUTINE pasa una petición de BIND desde el sistema principal a la aplicación LUA. La aplicación puede aceptar la petición BIND tal como llega, modificarla para intentar negociar los parámetros de BIND o rechazarla con el código de detección SNA adecuado.

Parámetros proporcionados

LUA proporciona los siguientes parámetros a la aplicación:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

lua_opcode

LUA_OPCODE_SLI_BIND_ROUTINE

lua_luname

Nombre en ASCII de la LU utilizada por la sesión.

lua_sid Identificador de la sesión.

lua_data_length

Longitud de RU de BIND proporcionada.

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene la RU de BIND proporcionada.

lua_th Los parámetros TH desde BIND.

lua_rh Los parámetros RH desde BIND.

Parámetros devueltos

Los parámetros devueltos por la aplicación dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta: BIND aceptado o negociado

Si la aplicación decide aceptar o negociar BIND, devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

SLI_BIND_ROUTINE

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene la RU de BIND proporcionada. Si la aplicación acepta BIND tal como llega, no deberá modificar el contenido del almacenamiento intermedio; si intenta negociar uno o más parámetros de BIND, deberá modificar los datos para establecer los parámetros adecuados en los valores preferidos.

Ejecución incorrecta: BIND rechazado

Si la aplicación decide rechazar BIND, devuelve los parámetros siguientes:

lua_prim_rc

LUA_NEGATIVE_RSP

lua_data_length

Longitud del código de detección SNA devuelto (en el parámetro *lua_data_ptr*).

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene el código de detección SNA asociado con el motivo de la aplicación para rechazar BIND.

Interacción con otros verbos

LUA llamará a esta rutina desde su proceso del verbo SLI_OPEN (después de que la aplicación haya emitido SLI_OPEN y antes de su retorno asíncrono).

Uso y restricciones

No existe ningún mecanismo de retorno asíncrono para las rutinas de ampliación de la aplicación. La rutina debe tener un retorno asíncrono.

SLI_SDT_ROUTINE

Este verbo se envía desde LUA a la aplicación (utilizando el punto de entrada de la rutina de ampliación SDT proporcionado por la aplicación en el verbo SLI_OPEN) y no desde la aplicación a LUA.

El verbo SLI_SDT_ROUTINE pasa una petición de SDT desde el sistema principal a la aplicación LUA. La aplicación puede responder a SDT con una respuesta o rechazarla con el código de detección SNA adecuado.

Parámetros proporcionados

LUA proporciona los siguientes parámetros a la aplicación:

lua_verb

LUA_VERB_SLI

lua_verb_length

Longitud en bytes del registro de verbo LUA.

lua_opcode

LUA_OPCODE_SLI_SDT_ROUTINE

lua_luname

Nombre en ASCII de la LU utilizada por la sesión.

lua_sid Identificador de la sesión.

lua_data_length

Longitud de RU de SDT proporcionada.

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene la RU de SDT proporcionada.

lua_th Los parámetros TH desde SDT.

lua_rh Los parámetros RH desde SDT.

Parámetros devueltos

Los parámetros devueltos por la aplicación dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta: respuesta a SDT

Si la aplicación decide aceptar SDT, devuelve los parámetros siguientes:

lua_prim_rc

LUA_OK

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene la RU de respuesta a SDT proporcionada.

Ejecución incorrecta: SDT rechazado

Si la aplicación decide rechazar SDT, devuelve los parámetros siguientes:

lua_prim_rc

LUA_NEGATIVE_RSP

lua_data_length

Longitud del código de detección SNA devuelto (en el parámetro *lua_data_ptr*).

lua_data_ptr

Puntero hacia el almacenamiento intermedio que contiene el código de detección SNA asociado con el motivo de la aplicación para rechazar SDT.

Interacción con otros verbos

LUA llamará a esta rutina desde su proceso del verbo SLI_OPEN (después de que la aplicación haya emitido SLI_OPEN y antes de su retorno asíncrono).

Uso y restricciones

No existe ningún mecanismo de retorno asíncrono para las rutinas de ampliación de la aplicación. La rutina debe tener un retorno asíncrono.

SLI_STSN_ROUTINE

Este verbo se envía desde LUA a la aplicación (utilizando el punto de entrada de la rutina de ampliación STSN proporcionado por la aplicación en el verbo SLI_OPEN) y no desde la aplicación a LUA.

El verbo SLI_STSN_ROUTINE pasa una petición de STSN desde el sistema principal a la aplicación LUA. La aplicación puede responder a STSN con una respuesta o rechazarla con el código de detección SNA adecuado.

Parámetros proporcionados

LUA proporciona los siguientes parámetros a la aplicación:

SLI_STSN_ROUTINE

lua_verb
LUA_VERB_SLI

lua_verb_length
Longitud en bytes del registro de verbo LUA.

lua_opcode
LUA_OPCODE_SLI_STSN_ROUTINE

lua_luname
Nombre en ASCII de la LU utilizada por la sesión.

lua_sid Identificador de la sesión.

lua_data_length
Longitud de RU de STSN proporcionada.

lua_data_ptr
Puntero hacia el almacenamiento intermedio que contiene la RU de STSN proporcionada.

lua_th Los parámetros TH desde STSN.

lua_rh Los parámetros RH desde STSN.

Parámetros devueltos

Los parámetros devueltos por la aplicación dependen de si el verbo ha finalizado correctamente; consulte los apartados siguientes.

Ejecución correcta: respuesta a STSN

Si la aplicación decide aceptar STSN, devuelve los parámetros siguientes:

lua_prim_rc
LUA_OK

lua_data_ptr
Puntero hacia el almacenamiento intermedio que contiene la RU de respuesta a STSN.

Ejecución incorrecta: STSN rechazado

Si la aplicación decide rechazar STSN, devuelve los parámetros siguientes:

lua_prim_rc
LUA_NEGATIVE_RSP

lua_data_length
Longitud del código de detección SNA devuelto (en el parámetro *lua_data_ptr*).

lua_data_ptr
Puntero hacia el almacenamiento intermedio que contiene el código de detección SNA asociado con el motivo de la aplicación para rechazar STSN.

Interacción con otros verbos

LUA llamará a esta rutina desde su proceso del verbo SLI_OPEN (después de que la aplicación haya emitido SLI_OPEN y antes de su retorno asíncrono).

Uso y restricciones

No existe ningún mecanismo de retorno asíncrono para las rutinas de ampliación de la aplicación. La rutina debe tener un retorno asíncrono.

Capítulo 6. Aplicación LUA de ejemplo

Este capítulo describe el programa de LUA de ejemplo de Communications Server para Linux **lsample.c**, escrito para el sistema operativo AIX o Linux, que ilustra la utilización de los verbos RUI de LUA. Este archivo se almacena en el directorio **/usr/lib/sna/samples** (AIX) o bien **/opt/ibm/sna/samples** (Linux).

Se facilita la siguiente información:

- Visión general del proceso de la aplicación
- Instrucciones para compilar, enlazar y ejecutar la aplicación

Visión general del proceso

La aplicación es un programa de emulación 3270 muy sencillo. Proporciona una pantalla sin formato con datos enviados desde el sistema principal (en las sesiones de LU y de SSCP), junto con los mensajes de estado (que indican si la aplicación está conectada a la sesión de LU o a la sesión de SSCP). Cuando se recibe una petición de respuesta concreta del sistema principal, se envía automáticamente una respuesta afirmativa. Los datos escritos por el usuario se envían al sistema principal, salvo dos pulsaciones especiales:

[(corchete izquierdo)

Cambia entre las sesiones de LU y de SSCP

] (corchete derecho)

Finaliza la aplicación

Una vez finalizado parte del proceso de inicialización, el programa consta básicamente de dos bucles principales: uno que lee los datos del sistema principal y otro que envía los datos suministrados por el usuario al sistema principal. Dichos bucles están implementados de la siguiente manera:

El **bucle de lectura** utiliza llamadas recursivas al verbo RUI_READ. La rutina de devolución de llamada realiza el siguiente proceso, que LUA llama cuando el verbo finaliza de forma asíncrona:

- Los datos tecleados aparecen en la pantalla.
- Se procesa la información de estado de sesión.
- Si se necesita una respuesta, se genera y se envía una respuesta afirmativa.
- Se vuelve a emitir el verbo RUI_READ para continuar el bucle.

Si el verbo finaliza de manera síncrona, al volver se llama explícitamente la misma rutina utilizada como rutina de devolución de llamada. Esto garantiza que se realiza el mismo proceso independientemente del tipo de retorno.

El **bucle de escritura** lee los datos del teclado. Si se realiza una de las dos pulsaciones, se lleva a cabo la acción correspondiente; de lo contrario, los datos de entrada se convierten a EBCDIC (código de intercambio decimal ampliado codificado en binario) (utilizando el verbo CSV CONVERT) para enviarlos al sistema principal en la sesión de LU o en la sesión de SSCP, según la sesión a la que la aplicación esté conectada. Los datos se envían utilizando el verbo RUI_WRITE; una vez más, se utiliza la rutina de devolución de llamada

Visión general del proceso

independientemente de si el verbo vuelve de forma asíncrona. El programa espera a que la rutina de devolución de llamada borre el semáforo antes de proseguir con el bucle.

Cuando el usuario realiza la pulsación de] para finalizar la aplicación, el programa interrumpe el bucle de escritura y emite el verbo RUI_TERM para finalizar la sesión. La sesión también finaliza si el bucle de lectura encuentra un código de retorno del verbo RUI_READ distinto de LUA_OK.

El flujo del programa se ilustra en el diagrama de la Figura 5:

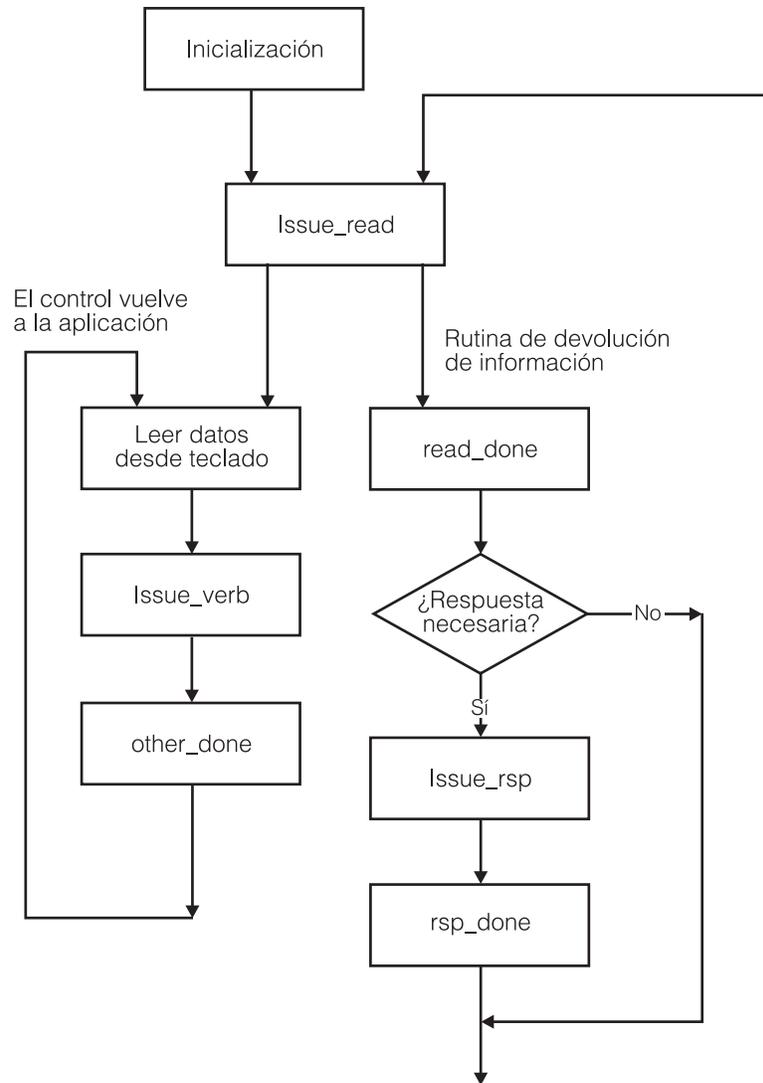


Figura 5. Flujo de programa para la aplicación LUA de ejemplo

Cómo probar la aplicación

Tras examinar el código fuente de la aplicación de ejemplo, es recomendable probarla. Deben llevarse a cabo los pasos siguientes:

1. Compruebe que tiene acceso a un sistema principal adecuado en el que pueda ejecutar la aplicación
2. Compile y enlace la aplicación

3. Configure Communications Server para Linux para utilizarlo con LUA (normalmente esta tarea la realizará el administrador del sistema)
4. Ejecute la aplicación

Estos pasos se describen de forma más detallada en los apartados siguientes.

Requisitos del sistema principal

Para ejecutar la aplicación de ejemplo, necesitará una LU en el sistema principal. Como la aplicación de ejemplo emula un terminal de pantalla 3270, la LU debe configurarse en el sistema principal como LU de pantalla 3270 (LU de tipo 2) como, por ejemplo, 3278 ó 3279. Al configurar la LU en Communications Server para Linux, se debe utilizar el número de LU asignado en el sistema principal.

Configuración de la aplicación de ejemplo

Communications Server para Linux se debe configurar para incluir la LU necesaria. Normalmente esta tarea la realiza el administrador del sistema. Se requieren los siguientes componentes:

- Un DLC, un puerto y una LS
- Una LU de tipo 0–3, con un número de LU que coincida con el de una LU adecuada del sistema principal

Estos componentes pueden tener el nombre que desee; la única información requerida por la aplicación es la LU que se utilizará para la sesión. Esta se pasa a la aplicación como un parámetro de línea de mandatos (nombre de LU) o como dos parámetros de línea de mandatos (nombre de PU y número de LU). Los siguientes elementos también son aplicables para la configuración de las LU:

- El número de LU configurado para esta LU en la configuración de Communications Server para Linux debe coincidir con el número de LU asignado en el sistema principal.
- Puede configurar una agrupación de LU para utilizarla con la aplicación y que contenga una o varias LU. A continuación, para acceder a la agrupación, puede proporcionar el nombre de la agrupación o el nombre de cualquier LU contenida en ella; se utilizará la LU utilizada menos recientemente de la agrupación.

Compilación y enlace de la aplicación de ejemplo

Para compilar y enlazar el programa para un sistema AIX o Linux, realice los pasos siguientes.

1. Copie el archivo `lsample.c` desde el directorio `/opt/ibm/sna/samples` a un directorio privado.
2. Para compilar y enlazar el programa para AIX, utilice el mandato siguiente:

```
cc -o lsample -I /usr/include/sna -bimport:/usr/lib/sna/lua.exp -bimport:/usr/lib/sna/csv.exp lsample.c
```

Para compilar y enlazar el programa para Linux, utilice el mandato siguiente:

```
gcc -o lsample -I /opt/ibm/sna/include -L /opt/ibm/sna/lib -llua -lsna -lcsv -lpLiS -lpthread lsample.c
```

Ejecución de la aplicación de ejemplo

En este apartado se supone que ha compilado y enlazado la aplicación de ejemplo según lo descrito en “Compilación y enlace de la aplicación de ejemplo”.

La aplicación de ejemplo utiliza tanto la interfaz de CSV como LUA; incluye llamadas al verbo CSV CONVERT para convertir datos suministrados por el usuario de ASCII a EBCDIC antes de enviarlos al sistema principal y para convertir datos recibidos del sistema principal a datos ASCII antes de mostrarlos

Cómo probar la aplicación

en la pantalla. Esta conversión utiliza una tabla de conversión definida por el usuario (Tabla G), que se almacena en un archivo del sistema Communications Server para Linux. Con el fuente de programa de aplicación de ejemplo de LUA, se proporciona un archivo adecuado, **luatblg.dat**, en el directorio **/usr/lib/sna/samples** (AIX) o **/opt/ibm/sna/samples** (Linux).

Para ejecutar la aplicación de ejemplo, siga estos pasos:

1. Asegúrese de que el software Communications Server para Linux se haya iniciado y de que la LS al sistema principal esté activa; si es necesario, póngase en contacto con el administrador del sistema.
2. Establezca la variable de entorno SNATBLG al nombre del archivo que contiene la tabla de conversión G. Incluya la vía completa del archivo si no se encuentra en el directorio actual.
3. Inicie la aplicación especificando uno de los siguientes mandatos:

lsample *nombrelu*

lsample *nombrepu númerolu*

En este ejemplo, *nombreLU* es el nombre de la LU configurada para esta aplicación (o el nombre de la agrupación de LU o de una LU de dicha agrupación).

En este ejemplo, *nombrePU* es el nombre de la PU propietaria de la LU requerida, y *númeroLU* es el número de LU (especificado como un número decimal).

La aplicación mostrará el mensaje **LU activa** cuando haya establecido correctamente una sesión con el sistema principal.

4. Especifique datos como lo haría normalmente para iniciar la sesión y acceder a las aplicaciones del sistema principal.
5. Para cambiar entre la sesión de LU y la sesión de SSCP, pulse la tecla [(corchete izquierdo) seguida de **Intro**.

La aplicación mostrará el mensaje **Sesión de LU** o **Sesión de SSCP** para indicar la sesión a la que está conectado. También cambia automáticamente cuando se recibe un mensaje BIND o UNBIND.

6. Cuando haya finalizado con las aplicaciones del sistema principal, siga los pasos que llevaría a cabo para finalizar las aplicaciones y terminar la sesión.
7. Para finalizar la aplicación, pulse la tecla] (corchete derecho) seguida de **Intro**.

La aplicación mostrará un mensaje de cierre **Closedown** seguido de un mensaje de finalización **Terminated** para indicar que ha finalizado la sesión con el sistema principal. Puede que también haya un mensaje de error de lectura "Read failed" con códigos de retorno que indiquen que el verbo RUI_TERM ha cancelado un verbo RUI_READ pendiente.

Apéndice A. Valores de código de retorno

Este apéndice lista todos los códigos de retorno posibles de la interfaz de LUA en orden numérico. Los valores se definen en el archivo de cabecera de LUA `lua_c.h` (para AIX o Linux) o `winlua.h` (para Windows).

Puede utilizar este apéndice como referencia para consultar el significado de un código de retorno recibido por la aplicación.

Códigos de retorno primarios

En las aplicaciones LUA se utilizan los códigos de retorno primarios siguientes.

| | |
|-------------------------------|--------|
| LUA_OK | 0x0100 |
| LUA_STATE_CHECK | 0x0200 |
| LUA_COMM_SUBSYSTEM_ABENDED | 0x03F0 |
| LUA_COMM_SUBSYSTEM_NOT_LOADED | 0x04F0 |
| LUA_INVALID_VERB_SEGMENT | 0x08F0 |
| LUA_SESSION_FAILURE | 0x0F00 |
| LUA_UNEXPECTED_DOS_ERROR | 0x11F0 |
| LUA_UNSUCCESSFUL | 0x1400 |
| LUA_STACK_TOO_SMALL | 0x15F0 |
| LUA_NEGATIVE_RSP | 0x1800 |
| LUA_CANCELLED | 0x2100 |
| LUA_IN_PROGRESS | 0x3000 |
| LUA_STATUS | 0x4000 |
| LUA_INVALID_VERB | 0xFFFF |

Códigos de retorno secundarios

En las aplicaciones LUA se utilizan los códigos de retorno secundarios siguientes.

| | |
|--------------------------------|------------|
| LUA_SEC_RC_OK | 0x00000000 |
| LUA_INVALID_LUNAME | 0x01000000 |
| LUA_BAD_SESSION_ID | 0x02000000 |
| LUA_DATA_TRUNCATED | 0x03000000 |
| LUA_BAD_DATA_PTR | 0x04000000 |
| LUA_DATA_SEG_LENGTH_ERROR | 0x05000000 |
| LUA_RESERVED_FIELD_NOT_ZERO | 0x06000000 |
| LUA_INVALID_POST_HANDLE | 0x07000000 |
| LUA_PURGED | 0x0C000000 |
| LUA_BID_VERB_SEG_ERROR | 0x0F000000 |
| LUA_NO_PREVIOUS_BID_ENABLED | 0x10000000 |
| LUA_NO_DATA | 0x11000000 |
| LUA_BID_ALREADY_ENABLED | 0x12000000 |
| LUA_VERB_RECORD_SPANS_SEGMENTS | 0x13000000 |
| LUA_INVALID_FLOW | 0x14000000 |
| LUA_NOT_ACTIVE | 0x15000000 |
| LUA_VERB_LENGTH_INVALID | 0x16000000 |
| LUA_REQUIRED_FIELD_MISSING | 0x19000000 |
| LUA_READY | 0x30000000 |
| LUA_NOT_READY | 0x31000000 |
| LUA_INIT_COMPLETE | 0x32000000 |
| LUA_SESSION_END_REQUESTED | 0x33000000 |
| LUA_NO_SLI_SESSION | 0x34000000 |
| LUA_SESSION_ALREADY_OPEN | 0x35000000 |
| LUA_INVALID_OPEN_INIT_TYPE | 0x36000000 |
| LUA_INVALID_OPEN_DATA | 0x37000000 |
| LUA_UNEXPECTED_SNA_SEQUENCE | 0x38000000 |
| LUA_NEG_RSP_FROM_BIND_ROUTINE | 0x39000000 |
| LUA_NEG_RSP_FROM_CRV_ROUTINE | 0x3A000000 |

Códigos de retorno secundarios

| | |
|--------------------------------|------------|
| LUA_NEG_RSP_FROM_STSN_ROUTINE | 0x3B000000 |
| LUA_CRV_ROUTINE_REQUIRED | 0x3C000000 |
| LUA_STSN_ROUTINE_REQUIRED | 0x3D000000 |
| LUA_INVALID_OPEN_ROUTINE_TYPE | 0x3E000000 |
| LUA_MAX_NUMBER_OF SENDS | 0x3F000000 |
| LUA_SEND_ON_FLOW_PENDING | 0x40000000 |
| LUA_INVALID_MESSAGE_TYPE | 0x41000000 |
| LUA_RECEIVE_ON_FLOW_PENDING | 0x42000000 |
| LUA_DATA_LENGTH_ERROR | 0x43000000 |
| LUA_CLOSE_PENDING | 0x44000000 |
| LUA_NEGATIVE_RSP_CHASE | 0x46000000 |
| LUA_NEGATIVE_RSP_SHUTC | 0x47000000 |
| LUA_NEGATIVE_RSP_RSHUTD | 0x48000000 |
| LUA_NO_RECEIVE_TO_PURGE | 0x4A000000 |
| LUA_CANCEL_COMMAND_RECEIVED | 0x4D000000 |
| LUA_RUI_WRITE_FAILURE | 0x4E000000 |
| LUA_INVALID_SESSION_TYPE | 0x4F000000 |
| LUA_SLI_BID_PENDING | 0x51000000 |
| LUA_SLI_PURGE_PENDING | 0x52000000 |
| LUA_PROCEDURE_ERROR | 0x53000000 |
| LUA_INVALID_SLI_ENCR_OPTION | 0x54000000 |
| LUA_RECEIVED_UNBIND | 0x55000000 |
| LUA_RECEIVED_UNBIND_HOLD | 0x56000000 |
| LUA_RECEIVED_UNBIND_NORMAL | 0x57000000 |
| LUA_SLI_LOGIC_ERROR | 0x7F000000 |
| LUA_TERMINATED | 0x80000000 |
| LUA_NO_RUI_SESSION | 0x81000000 |
| LUA_DUPLICATE_RUI_INIT | 0x82000000 |
| LUA_INVALID_PROCESS | 0x83000000 |
| LUA_API_MODE_CHANGE | 0x85000000 |
| LUA_COMMAND_COUNT_ERROR | 0x87000000 |
| LUA_NO_READ_TO_PURGE | 0x88000000 |
| LUA_MULTIPLE_WRITE_FLOWS | 0x89000000 |
| LUA_DUPLICATE_READ_FLOW | 0x8A000000 |
| LUA_DUPLICATE_WRITE_FLOW | 0x8B000000 |
| LUA_LINK_NOT_STARTED | 0x8C000000 |
| LUA_INVALID_ADAPTER | 0x8D000000 |
| LUA_ENCR_DECR_LOAD_ERROR | 0x8E000000 |
| LUA_ENCR_DECR_PROC_ERROR | 0x8F000000 |
| LUA_INVALID_PUNAME | 0x90000000 |
| LUA_UNAUTHORIZED_ACCESS | 0x90020000 |
| LUA_INVALID_LUNUMBER | 0x91000000 |
| LUA_INVALID_FORMAT | 0x92000000 |
| LUA_DUPLICATE_RUI_REINIT | 0x93000000 |
| LUA_REINIT_INVALID | 0x94000000 |
| LUA_TCPCV_LENGTH_INVALID | 0x95000000 |
| LUA_LINK_NOT_STARTED_RETRY | 0x95FF0000 |
| LUA_NEG_RSP_FROM_SDT_ROUTINE | 0x96000000 |
| LUA_NEG_NOTIFY_RSP | 0xBE000000 |
| LUA_RUI_LOGIC_ERROR | 0xBF000000 |
| LUA_COBOL_NOT_SUPPORTED | 0xC0000000 |
| LUA_DUPLICATE_RUI_INIT_PRIMARY | 0xC2000000 |
| LUA_LU_INOPERATIVE | 0xFF000000 |

Los siguientes códigos de retorno secundarios son códigos de detección de SNA. Se listan en el orden de bytes estándar utilizado por LUA y en el orden de bytes utilizado para los códigos de detección de SNA en los manuales de consulta de SNA.

| | | |
|-------------------------------|------------|---------------------------|
| LUA_RESOURCE_NOT_AVAILABLE | 0x00000108 | (detección SNA 0801 0000) |
| LUA_RU_DATA_ERROR | 0x00000110 | (detección SNA 1001 0000) |
| LUA_INCORRECT_SEQUENCE_NUMBER | 0x00000120 | (detección SNA 2001 0000) |
| LUA_INVALID_SC_OR_NC_RH | 0x00000140 | (detección SNA 4001 0000) |
| LUA_RU_LENGTH_ERROR | 0x00000210 | (detección SNA 1002 0000) |
| LUA_CHAINING_ERROR | 0x00000220 | (detección SNA 2002 0000) |
| LUA_FUNCTION_NOT_SUPPORTED | 0x00000310 | (detección SNA 1003 0000) |
| LUA_BRACKET | 0x00000320 | (detección SNA 2003 0000) |

Códigos de retorno secundarios

| | | |
|--------------------------------|------------|---------------------------|
| LUA_BB_NOT_ALLOWED | 0x00000340 | (detección SNA 4003 0000) |
| LUA_NAU_INOPERATIVE | 0x00000380 | (detección SNA 8003 0000) |
| LUA_DIRECTION | 0x00000420 | (detección SNA 2004 0000) |
| LUA_EB_NOT_ALLOWED | 0x00000440 | (detección SNA 4004 0000) |
| LUA_SESSION_LIMIT_EXCEEDED | 0x00000508 | (detección SNA 0805 0000) |
| LUA_DATA_TRAFFIC_RESET | 0x00000520 | (detección SNA 2005 0000) |
| LUA_NO_SESSION | 0x00000580 | (detección SNA 8005 0000) |
| LUA_DATA_TRAFFIC QUIESCED | 0x00000620 | (detección SNA 2006 0000) |
| LUA_EXCEPTION_RSP_NOT_ALLOWED | 0x00000640 | (detección SNA 4006 0000) |
| LUA_CATEGORY_NOT_SUPPORTED | 0x00000710 | (detección SNA 1007 0000) |
| LUA_DATA_TRAFFIC_NOT_RESET | 0x00000720 | (detección SNA 2007 0000) |
| LUA_DEFINITE_RSP_NOT_ALLOWED | 0x00000740 | (detección SNA 4007 0000) |
| LUA_NO_BEGIN_BRACKET | 0x00000820 | (detección SNA 2008 0000) |
| LUA_PACING_NOT_SUPPORTED | 0x00000840 | (detección SNA 4008 0000) |
| LUA_MODE_INCONSISTENCY | 0x00000908 | (detección SNA 0809 0000) |
| LUA_SC_PROTOCOL_VIOLATION | 0x00000920 | (detección SNA 2009 0000) |
| LUA_CD_NOT_ALLOWED | 0x00000940 | (detección SNA 4009 0000) |
| LUA_IMMEDIATE_REQ_MODE_ERROR | 0x00000A20 | (detección SNA 200A 0000) |
| LUA_NO_RESPONSE_NOT_ALLOWED | 0x00000A40 | (detección SNA 400A 0000) |
| LUA_BRACKET_RACE_ERROR | 0x00000B08 | (detección SNA 800B 0000) |
| LUA_QUEUED_RESPONSE_ERROR | 0x00000B20 | (detección SNA 200B 0000) |
| LUA_CHAINING_NOT_SUPPORTED | 0x00000B40 | (detección SNA 400B 0000) |
| LUA_ERP_SYNC_EVENT_ERROR | 0x00000C20 | (detección SNA 200C 0000) |
| LUA_BRACKETS_NOT_SUPPORTED | 0x00000C40 | (detección SNA 400C 0000) |
| LUA_RSP_BEFORE_SENDING_REQ | 0x00000D20 | (detección SNA 200D 0000) |
| LUA_CD_NOT_SUPPORTED | 0x00000D40 | (detección SNA 400D 0000) |
| LUA_RSP_CORRELATION_ERROR | 0x00000E20 | (detección SNA 200E 0000) |
| LUA_RSP_PROTOCOL_ERROR | 0x00000F20 | (detección SNA 200F 0000) |
| LUA_INCORRECT_USE_OF_FI | 0x00000F40 | (detección SNA 400F 0000) |
| LUA_ALTERNATE_CODE_NOT_SUPPORT | 0x00001040 | (detección SNA 4001 0000) |
| LUA_INCORRECT_RU_CATEGORY | 0x00001140 | (detección SNA 4011 0000) |
| LUA_INSUFFICIENT_RESOURCES | 0x00001208 | (detección SNA 0812 0000) |
| LUA_INCORRECT_REQUEST_CODE | 0x00001240 | (detección SNA 4012 0000) |
| LUA_BB_REJECT_NO_RTR | 0x00001308 | (detección SNA 0813 0000) |
| LUA_INCORRECT_SPEC_OF_SDI_RTI | 0x00001340 | (detección SNA 4013 0000) |
| LUA_BB_REJECT_RTR | 0x00001408 | (detección SNA 0814 0000) |
| LUA_INCORRECT_DR1I_DR2I_ERI | 0x00001440 | (detección SNA 4014 0000) |
| LUA_INCORRECT_USE_OF_QRI | 0x00001540 | (detección SNA 4015 0000) |
| LUA_INCORRECT_USE_OF EDI | 0x00001640 | (detección SNA 4016 0000) |
| LUA_INCORRECT_USE_OF_PDI | 0x00001740 | (detección SNA 4017 0000) |
| LUA_RECEIVER_IN_TRANSMIT_MODE | 0x00001B08 | (detección SNA 081B 0000) |
| LUA_REQUEST_NOT_EXECUTABLE | 0x00001C08 | (detección SNA 081C 0000) |
| LUA_INVALID_SESSION_PARAMETERS | 0x00002108 | (detección SNA 0821 0000) |
| LUA_UNIT_OF_WORK_ABORTED | 0x00002408 | (detección SNA 0824 0000) |
| LUA_FM_FUNCTION_NOT_SUPPORTED | 0x00002608 | (detección SNA 0826 0000) |
| LUA_LU_COMPONENT_DISCONNECTED | 0x00003108 | (detección SNA 0831 0000) |
| LUA_INVALID_PARAMETER_FLAGS | 0x00003308 | (detección SNA 0833 0000) |
| LUA_INVALID_PARAMETER | 0x00003508 | (detección SNA 0835 0000) |
| LUA_CRYPTOGRAPHY_INOPERATIVE | 0x00004808 | (detección SNA 0848 0000) |
| LUA_REQ_RESOURCES_NOT_AVAIL | 0x00004B08 | (detección SNA 084B 0000) |
| LUA_SSCP_LU_SESSION_NOT_ACTIVE | 0x00005708 | (detección SNA 0857 0000) |
| LUA_SYNC_EVENT_RESPONSE | 0x00006708 | (detección SNA 0867 0000) |
| LUA_SESSION_SERVICE_PATH_ERROR | 0x00007D08 | (detección SNA 087D 0000) |
| LUA_NEGOTIABLE_BIND_ERROR | 0x01003508 | (detección SNA 0835 0001) |
| LUA_REC_CORR_TABLE_FULL | 0x01007808 | (detección SNA 0878 0001) |
| LUA_NON_UNIQ_ID | 0x011000C0 | (detección SNA C000 1001) |
| LUA_INV_NAU_ADDR | 0x012000C0 | (detección SNA C000 2001) |
| LUA_BIND_FM_PROFILE_ERROR | 0x02003508 | (detección SNA 0835 0002) |
| LUA_SSCP_PLU_SESS_NOT_ACTIVE | 0x02005708 | (detección SNA 0857 0002) |
| LUA_SEND_CORR_TABLE_FULL | 0x02007808 | (detección SNA 0878 0002) |
| LUA_NON_UNIQ_NAU_AD | 0x021000C0 | (detección SNA C000 1002) |
| LUA_INV_ADPT_NUM | 0x022000C0 | (detección SNA C000 2002) |
| LUA_BIND_TS_PROFILE_ERROR | 0x03003508 | (detección SNA 0835 0003) |
| LUA_SSCP_SLU_SESS_INACT | 0x03005708 | (detección SNA 0857 0003) |
| LUA_SLU_SESSION_LIMIT_EXCEEDED | 0x0A000508 | (detección SNA 0805 000A) |
| LUA_BIND_LU_TYPE_ERROR | 0x0E003508 | (detección SNA 0835 000E) |
| LUA_HDX_BRACKET_STATE_ERROR | 0x21010510 | (detección SNA 1005 0121) |

Códigos de retorno secundarios

| | | |
|-----------------------------|------------|---------------------------|
| LUA_RESPONSE_ALREADY_SENT | 0x22010510 | (detección SNA 1005 0122) |
| LUA_EXR_SENSE_INCORRECT | 0x23010510 | (detección SNA 1005 0123) |
| LUA_RESPONSE_OUT_OF_ORDER | 0x24010510 | (detección SNA 1005 0124) |
| LUA_CHASE_RESPONSE_REQUIRED | 0x25010510 | (detección SNA 1005 0125) |

Apéndice B. Avisos

La presente información se ha desarrollado para productos y servicios ofrecidos en EE.UU. Es posible que IBM no ofrezca en otros países los productos, los servicios o las características que se describen en este documento. Consulte al representante local de IBM para obtener información sobre los productos y servicios actualmente disponibles en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar ese producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad de intelectual de IBM. Sin embargo, la evaluación y la verificación del funcionamiento de cualquier producto, programa o servicio que no sea de IBM son responsabilidad del usuario.

IBM puede tener patentes o solicitudes de patente en tramitación que incluyan el tema principal descrito en este documento. La entrega de este documento no le concede ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para cualquier consulta sobre licencias relacionada con la información sobre DBCS (juego de caracteres de doble byte), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe su consulta por escrito a la siguiente dirección:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país en el que estas disposiciones sean contrarias a la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O ADECUACIÓN PARA UN FIN DETERMINADO. Algunos estados no permiten la renuncia de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable a su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en la información aquí contenida; estos cambios se incorporarán en las nuevas ediciones de la publicación. IBM puede realizar, en cualquier momento y sin previo aviso, mejoras y/o cambios en los productos y/o programas descritos en esta publicación.

Las referencias hechas en esta información a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen en modo alguno un aval de estos sitios Web. La información contenida en estos sitios Web

no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de estos sitios Web.

IBM puede utilizar o distribuir la información que se le proporcione del modo que considere apropiado, sin incurrir por ello en ninguna obligación con el remitente.

Los propietarios de licencias de este programa que deseen disponer de información acerca del mismo con el fin de permitir: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información que se ha intercambiado, deben ponerse en contacto con:

IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195
EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, que pueden incluir, en algunos casos, el pago de una cuota.

IBM proporciona el programa bajo licencia que se describe en esta información y todo el material bajo licencia disponible para él conforme a los términos del contrato de cliente de IBM, el acuerdo internacional de licencia de programas de IBM o cualquier acuerdo equivalente entre las partes.

Los datos sobre rendimiento que este documento contiene se determinaron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse efectuado en sistemas a nivel de desarrollo y no existe ninguna garantía de que estas mediciones vayan a ser las mismas en sistemas disponibles de forma general. Además, es posible que algunas medidas se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha comprobado estos productos y no puede confirmar con precisión su rendimiento, su compatibilidad o cualquier otra reclamación relacionada con productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben formularse a sus respectivos proveedores.

Esta información contiene ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones empleados por una empresa es totalmente fortuita.

LICENCIA DE COPYRIGHT: Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones de acuerdo con la interfaz de programación de aplicaciones para la plataforma operativa en la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por tanto, IBM no puede

asegurar ni implicar la fiabilidad, utilidad o función de estos programas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin realizar pago alguno a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a las interfaces de programación de aplicaciones de IBM.

Cada copia total o parcial de estos programas de ejemplo o de cualquier trabajo derivado deben incluir un aviso de copyright como el siguiente: ® (nombre de la empresa) (año). Partes de este código se han obtenido de Programas de ejemplo de IBM Corp. ® Copyright IBM Corp. 2000, 2005, 2006. Reservados todos los derechos.

Marcas registradas

Los términos siguientes son marcas registradas de IBM Corporation en Estados Unidos y/o en otros países:

| | |
|----------------------------------|----------------------------------|
| Advanced Peer-to-Peer Networking | Power5 |
| AIX | pSeries |
| Application System/400 | S/390 |
| AS/400 | SP |
| CICS | System/370 |
| IBM | System/390 |
| MQSeries | SAA |
| MVS | Systems Application Architecture |
| MVS/ESA | VTAM |
| MVS/XA | WebSphere |
| NetView | z/OS |
| OpenPower | z9 |
| OS/2 | zSeries |

Los términos siguientes son marcas registradas de otras empresas:

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en EE.UU. y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en EE.UU. y/o en otros países.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus subsidiarias en EE.UU. y en otros países.

UNIX es una marca registrada de The Open Group en EE.UU. y en otros países.

Linux es una marca registrada de Linus Torvalds en EE.UU. y/o en otros países.

Otros nombres de compañías, productos y servicios pueden ser marcas registradas de otras empresas.

Bibliografía

Las siguientes publicaciones de IBM proporcionan información sobre los temas descritos en esta biblioteca. Las publicaciones se dividen en las siguientes grandes áreas temáticas:

- Communications Server para Linux, Versión 6.2.2
- Systems Network Architecture (SNA)
- Configuración del sistema principal
- z/OS Communications Server
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- X.25
- Advanced Program-to-Program Communication (APPC)
- Programación
- Otros temas de gestión de redes de IBM

Se proporcionan breves descripciones de los manuales de la biblioteca de Communications Server para Linux. Para los demás manuales, sólo se muestran el título, el número de pedido y, en algunos casos, el título abreviado que se utiliza en el texto del presente manual.

Publicaciones de Communications Server para Linux Versión 6.2.2

La biblioteca de Communications Server para Linux consta de los siguientes manuales. Además, se proporcionan versiones en copia software de estos documentos en el CD-ROM. Consulte la publicación *IBM Communications Server para Linux, Guía de iniciación rápida* para obtener información sobre cómo acceder a los archivos de copia software del CD-ROM. Para instalar en el sistema estos manuales en copia software, necesitará 9–15 MB de espacio de disco duro (según la versión de idioma que instale).

- *IBM Communications Server para Linux, Guía de iniciación rápida* (GC10-9852-01)
Este manual ofrece una presentación general de Communications Server para Linux, incluyendo información sobre las características de red soportadas, la instalación, la configuración y el funcionamiento.
- *IBM Communications Server para Linux, Guía de administración* (SC10-9853-01)
Este manual proporciona una visión general de SNA y de Communications Server para Linux e información sobre la configuración y el funcionamiento de Communications Server para Linux.
- *IBM Communications Server for Linux, Administration Command Reference* (SC31-6770-02)
Este manual proporciona información sobre los mandatos de SNA y Communications Server para Linux.
- *IBM Communications Server para Linux, Guía del programador para CPI-C* (SC10-9861-01)
Este manual proporciona información para programadores expertos en “C” o Java sobre cómo escribir programas de transacciones SNA utilizando la API de CPI Communications de Communications Server para Linux.
- *IBM Communications Server para Linux, Guía del programador para APPC* (SC10-9854-01)

Este manual contiene la información necesaria para desarrollar programas de aplicación mediante APPC (comunicación avanzada programa a programa).

- *IBM Communications Server para Linux, Guía del programador para LUA* (SC10-9855-01)

Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la interfaz de programas de aplicación de LU (LUA) convencional.

- *IBM Communications Server for Linux, CSV Programmer's Guide* (SC31-6775-02)

Este manual contiene la información necesaria para desarrollar programas de aplicación utilizando la interfaz de programas de aplicación (API) de CSV (Common Service Verbs).

- *IBM Communications Server for Linux, MS Programmer's Guide* (SC31-67770-02)

Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la API de MS (Management Services).

- *IBM Communications Server for Linux, NOF Programmer's Guide* (SC31-6778-02)

Este manual contiene la información necesaria para desarrollar aplicaciones utilizando la API de NOF (Node Operator Facility).

- *IBM Communications Server para Linux, Guía de diagnósticos* (SC11-3348-01)

Este manual proporciona información sobre la resolución de problemas en redes SNA.

- *IBM Communications Server for Linux, APPC Application Suite User's Guide* (SC31-6772-02)

Este manual proporciona información sobre las aplicaciones APPC utilizadas con Communications Server para Linux.

- *IBM Communications Server para Linux, Glossary* (GC31-6780-02)

Este manual proporciona una lista completa de los términos y las definiciones que se utilizan en toda la biblioteca de IBM Communications Server para Linux.

Publicaciones de SNA (Systems Network Architecture)

Los manuales siguientes contienen información sobre las redes SNA:

- *Systems Network Architecture: Format and Protocol Reference Manual—Architecture Logic for LU Type 6.2* (SC30-3269)
- *Systems Network Architecture: Formats* (GA27-3136)
- *Systems Network Architecture: Guide to SNA Publications* (GC30-3438)
- *Systems Network Architecture: Network Product Formats* (LY43-0081)
- *Systems Network Architecture: Technical Overview* (GC30-3073)
- *Systems Network Architecture: APPN Architecture Reference* (SC30-3422)
- *Systems Network Architecture: Sessions between Logical Units* (GC20-1868)
- *Systems Network Architecture: LU 6.2 Reference—Peer Protocols* (SC31-6808)
- *Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2* (GC30-3084)
- *Systems Network Architecture: 3270 Datastream Programmer's Reference* (GA23-0059)
- *Networking Blueprint Executive Overview* (GC31-7057)
- *Systems Network Architecture: Management Services Reference* (SC30-3346)

Publicaciones de configuración de sistema principal

Los manuales siguientes contienen información sobre la configuración de sistemas principales:

- *ES/9000, ES/3090 IOCP User's Guide Volume A04* (GC38-0097)
- *3174 Establishment Controller Installation Guide* (GG24-3061)
- *3270 Information Display System 3174 Establishment Controller: Planning Guide* (GA27-3918)
- *OS/390 Hardware Configuration Definition (HCD) User's Guide* (SC28-1848)

Publicaciones de z/OS Communications Server

Los manuales siguientes contienen información sobre z/OS Communications Server:

- *z/OS V1R7 Communications Server: SNA Network Implementation Guide* (SC31-8777)
- *z/OS V1R7 Communications Server: SNA Diagnostics* (Vol 1: GC31-6850, Vol 2: GC31-6851)
- *z/OS V1R6 Communications Server: Resource Definition Reference* (SC31-8778)

Publicaciones sobre TCP/IP

Los manuales siguientes contienen información sobre el protocolo de red TCP/IP (Transmission Control Protocol/Internet Protocol - Protocolo de control de transmisiones/Protocolo Internet):

- *z/OS V1R7 Communications Server: IP Configuration Guide* (SC31-8775)
- *z/OS V1R7 Communications Server: IP Configuration Reference* (SC31-8776)
- *z/VM V5R1 TCP/IP Planning and Customization* (SC24-6125)

Publicaciones sobre X.25

Los manuales siguientes contienen información sobre el protocolo de red X.25:

- *Communications Server for OS/2 Version 4 X.25 Programming* (SC31-8150)

Publicaciones de APPC

Los manuales siguientes contienen información sobre APPC (comunicación avanzada programa a programa):

- *APPC Application Suite V1 User's Guide* (SC31-6532)
- *APPC Application Suite V1 Administration* (SC31-6533)
- *APPC Application Suite V1 Programming* (SC31-6534)
- *APPC Application Suite V1 Online Product Library* (SK2T-2680)
- *APPC Application Suite Licensed Program Specifications* (GC31-6535)
- *z/OS V1R2.0 Communications Server: APPC Application Suite User's Guide* (SC31-8809)

Publicaciones de programación

Los manuales siguientes contienen información sobre programación:

- *Common Programming Interface Communications CPI-C Reference* (SC26-4399)
- *Communications Server for OS/2 Version 4 Application Programming Guide* (SC31-8152)

Otras publicaciones de gestión de red de IBM

Los manuales siguientes contienen información sobre otros temas relacionados con Communications Server para Linux:

- *SDLC Concepts* (GA27-3093)
- *Local Area Network Concepts and Products: LAN Architecture* (SG24-4753)
- *Local Area Network Concepts and Products: LAN Adapters, Hubs and ATM* (SG24-4754)
- *Local Area Network Concepts and Products: Routers and Gateways* (SG24-4755)
- *Local Area Network Concepts and Products: LAN Operating Systems and Management* (SG24-4756)
- *IBM Network Control Program Resource Definition Guide* (SC30-3349)

Índice

A

- ACTLU 8
- acuse de recibo de cortesía 39, 41
- agrupaciones de LU 5, 43
- aplicación de ejemplo
 - configuración 159
 - ejecutar 159
 - probar 158
 - requisitos del sistema principal 159
 - visión general del proceso 157
- aplicaciones AIX
 - compilar y enlazar 44
- aplicaciones Linux
 - compilar y enlazar 44

B

- BIND 8

C

- CANCEL 40, 42
- códigos de detección SNA 37, 46
 - valores 162
- códigos de retorno
 - primarios 161
 - secundarios 161
- códigos de retorno primarios 161
- códigos de retorno secundarios 161
- compatibilidad, con IBM OS/2 Extended Edition 45
- compilar aplicaciones AIX 44
- compilar aplicaciones Linux 44
- compilar y enlazar 44
- componentes SNA necesarios para las comunicaciones
 - LUA 2, 4
- conceptos de LUA 1
- Consideraciones sobre el entorno Linux 43
- consideraciones sobre el entorno Windows 44
- conversión ASCII a EBCDIC 159
- conversión EBCDIC a ASCII 159
- CSV CONVERT, verbo 159

D

- definición de LUA 1

E

- ejemplo de secuencia de comunicaciones LUA 8
- eliminar elementos innecesarios 40, 42
- enlazar aplicaciones AIX 44
- enlazar aplicaciones Linux 44
- establecer la sesión de SSCP 8
- estructura de datos común 48, 49
- estructura de datos específica 54
- estructura de datos LUA_VERB_RECORD 48

F

- finalización asíncrona de verbos 7, 15
- finalización síncrona del verbo 15, 22, 32
- flujo normal 4
- flujo rápido 4

I

- información de configuración 5, 42
- información de SNA 36
 - RUI primaria 40
- INITSELF 8
- llamada a GetLuaReturnCode 27
- llamadas de función para LUA 16

M

- manejador de ventanas 7
- mensajes SNA, relación con los verbos LUA 9

N

- NOTIFY 8

P

- parámetros BIND, negociar 36, 37
- parámetros reservados 45, 47, 59, 107
- portabilidad a otros entornos 45
- Publicaciones de SNA (arquitectura de red de sistemas) xiii
- punto de entrada 14
 - LUA 14
 - Windows 16
- punto de entrada RUI 14
- Punto de entrada SLI 14

R

- resumen de verbos LUA
 - RUI 5
 - SLI 6
- resumen de verbos RUI 5
- resumen de verbos SLI 6
- ritmo 38
 - RUI primaria 41
- RU 1, 2
- RUI 2
- RUI_BID
 - interacción con otros verbos 65
 - parámetros devueltos 60
 - parámetros proporcionados 59
 - uso y restricciones 65
- RUI_INIT
 - interacción con otros verbos 72
 - parámetros proporcionados 66
 - uso y restricciones 72
- RUI_INIT_PRIMARY
 - interacción con otros verbos 76

- RUI_INIT_PRIMARY (*continuación*)
 - parámetros proporcionados 73
 - uso y restricciones 77
- RUI_PURGE
 - interacción con otros verbos 81
 - parámetros devueltos 78
 - parámetros proporcionados 77
- RUI_READ
 - interacción con otros verbos 89
 - parámetros devueltos 83
 - parámetros proporcionados 82
 - uso y restricciones 89
- RUI_REINIT
 - interacción con otros verbos 93
 - parámetros devueltos 91
 - parámetros proporcionados 90
 - uso y restricciones 93
- RUI_TERM
 - interacción con otros verbos 98
 - parámetros devueltos 95
 - parámetros proporcionados 94
- RUI_WRITE
 - interacción con otros verbos 105
 - parámetros devueltos 100
 - parámetros proporcionados 98
 - uso y restricciones 105
- RUI y SLI, comparación 2
- rutina de devolución de llamada 7, 16

S

- SDT 8
- segmentación 39
 - RUI primaria 41
- sesión de LU 3
- sesión de SSCP 3
- sesión PU-SSCP 3
- SLI 2
- SLI_BID
 - interacción con otros verbos 114
 - parámetros devueltos 108
 - parámetros proporcionados 107
 - uso y restricciones 114
- SLI_BIND_ROUTINE
 - interacción con otros verbos 154
 - parámetros devueltos 153, 154
 - parámetros proporcionados 153
 - uso y restricciones 154
- SLI_CLOSE
 - interacción con otros verbos 119
 - parámetros devueltos 116
 - parámetros proporcionados 115
 - uso y restricciones 120
- SLI_OPEN
 - interacción con otros verbos 129
 - parámetros devueltos 125
 - parámetros proporcionados 121
 - uso y restricciones 130
- SLI_PURGE
 - interacción con otros verbos 134
 - parámetros devueltos 131
 - parámetros proporcionados 130
- SLI_RECEIVE
 - interacción con otros verbos 143
 - parámetros devueltos 136
 - parámetros proporcionados 134
 - uso y restricciones 143

- SLI_SDT_ROUTINE
 - interacción con otros verbos 155
 - parámetros devueltos 155
 - parámetros proporcionados 154
 - uso y restricciones 155
- SLI_SEND
 - interacción con otros verbos 152
 - parámetros devueltos 146
 - parámetros proporcionados 144
 - uso y restricciones 152
- SLI_STSN_ROUTINE
 - interacción con otros verbos 156
 - parámetros devueltos 156
 - parámetros proporcionados 155
 - uso y restricciones 156
- SLI y RUI, comparación 2
- SSCP 3
- subproceso 43

T

- tipos de LU 1

U

- UNBIND 8

V

- varios procesos 43
- VCB
 - estructura 14
 - estructura de datos común 47
 - estructura de datos específica 47
 - formato 47
- verbo LUA, emitir 33



Número de Programa: 5724-i33, 5724-i34

SC10-9855-01

