SIP Server Platform Message Layer

Reference Guide

RADVISION

Delivering the Visual Experience

NOTICE

© 2001-2008 RADVISION Ltd. All intellectual property rights in this publication are owned by RADVISION Ltd. and are protected by United States copyright laws, other applicable copyright laws and international treaty provisions. RADVISION Ltd. retains all rights not expressly granted.

This publication is RADVISION confidential. No part of this publication may be reproduced in any form whatsoever or used to make any derivative work without prior written approval by RADVISION Ltd.

No representation of warranties for fitness for any purpose other than what is specifically mentioned in this guide is made either by RADVISION Ltd. or its agents.

RADVISION Ltd. reserves the right to revise this publication and make changes without obligation to notify any person of such revisions or changes. RADVISION Ltd. may make improvements or changes in the product(s) and/or the program(s) described in this documentation at any time.

If there is any software on removable media described in this publication, it is furnished under a license agreement included with the product as a separate document. If you are unable to locate a copy, please contact RADVISION Ltd. and a copy will be provided to you.

Unless otherwise indicated, RADVISION registered trademarks are registered in the United States and other territories. All registered trademarks recognized.

For further information contact RADVISION or your local distributor or reseller.

RADVISION SIP Server Platform Message Layer Reference Guide version 3.5, March 2008

Publication 8

http://www.radvision.com

CONTENTS

About This Manual

Documentation ix

1 *Message Object Functions*

Control Functions 4
Get and Set Functions 12

2 *Address Functions*

Control Functions 70
Get and Set Functions 89
Address Comparison Functions 95
SIP URL Get and Set Functions 97
Absolute URI Get and Set Functions 139
TEL URI Get and Set Functions 144
Diameter Get and Set Functions 161

3 *Allow Header Functions*

Control Functions 176
Get and Set Functions 184

4 *Allow-Events Header Functions*

Control Functions 194

Get and Set Functions 202

5 *Authentication Header Functions*

Control Functions 216
Get and Set Functions 224

6 *Authentication-Info Header Functions*

Control Functions 264
Get and Set Functions 272

7 *Authorization Header Functions*

Control Functions 292
Get and Set Functions 300

8 *Contact Header Functions*

Control Functions 344
Get and Set Functions 353

9 *Content-Disposition Header Functions*

Control Functions 388
Get and Set Functions 397

10 *Content-Type Header Functions*

Control Functions 412
Get and Set Functions 420

11 *Content-ID Header Functions*

Control Functions 452
Get and Set Functions 460

12 *CSeq Header Functions*

Control Functions 469
Get and Set Functions 478

13 *Date Header Functions*

Control Functions 490
Get and Set Functions 500

14 *Event Header Functions*

Control Functions 520
Get and Set Functions 529

15 *Expires Header Functions*

Control Functions 548
Get and Set Functions 557

16 *Min-SE Header Functions*

Control Functions 568
Get and Set Functions 576

17 *Other Header Functions*

Control Functions 586
Get and Set Functions 593

18 *Party Header Functions*

Control Functions 606
Get and Set Functions 618

19 *RAck Header Functions*

Control Functions 638
Get and Set Functions 647

20 *Refer-To Header Functions*

Control Functions 658
Get and Set Functions 666

21 *Referred-By Header Functions*

Control Functions 682
Get and Set Functions 690

22 *Retry-After Header Functions*

Control Functions 708
Get and Set Functions 716

23 *Route-Hop Header Functions*

Control Functions 734
Get and Set Functions 742

24 *RSeq Header Functions*

Control Functions 760
Get and Set Functions 768

25 *Session-Expires Header Functions*

Control Functions 778
Get and Set Functions 786

26 *Subscription-State Header Functions*

Control Functions 800
Get and Set Functions 808

27 *Body Functions*

Control Functions 828
Get and Set Functions 837

28 *Via Header Functions*

Control Functions 852
Get and Set Functions 860

29 *Functions for All Headers*

Control Functions 898

30 *Message Type Definitions*

Handle Type Definitions 906
String Name Type Definitions 948
Message API Type Definitions 981

Index 1

ABOUT THIS MANUAL

The RADVISION SIP Server Platform Message Layer is for developers of SIP Server applications. This Reference Guide lists the Application Programming Interfaces (API) functions of the SIP Server Toolkit. Each function has a general description, syntax description and other important information. The Reference Guide also contains the different enumerations and structure types that are defined in the API.

DOCUMENTATION

See the *SIP Server Platform Release Notes* for a complete list of documentation for the SIP Server Platform and SIP Server Platform Add-on Modules.

DOCUMENTATION NOTES

- The Reference Guide in PDF format has been divided into the *RADVISION SIP Server Platform Reference Guide* and *RADVISION SIP Server Platform Message Layer Reference Guide* for easier handling. The *RADVISION SIP Server Platform Online Reference Guide* in HTML and CHM formats contains the content of both books.
- All references to High Availability, OMAP, Accounting and Events in this manual are relevant only to the optional SIP Server add-on modules.

ADDITIONAL PRODUCTS

The SIP Server Platform is part of the RADVISION SIP product family which also includes these products:

RADVISION IMS SIP Toolkit (Stack)

The RADVISION IMS SIP Toolkit consists of SIP, SDP and RTP/RTCP Stacks, a suite of APIs, sample code and user documentation for developers who wish to build SIP applications. The Toolkit includes IMS support. For more information, see the *RADVISION IMS SIP Toolkit Programmer Guide* and the *RADVISION IMS SIP Add-on Module Programmer and Reference Guide*.

ProLab™ SIP Test Manager

The ProLab™ SIP Test Manager is a highly scalable, feature-rich testing system that is suitable for use in different stages of the product development cycle. The ProLab™ SIP Test Manager can emulate a group of synchronized SIP entities, such as SIP user agents (UAs) and SIP Servers. For more information, see the *RADVISION ProLab™ SIP Test Manager User Guide.*

Multi-media Terminal Framework

The RADVISION Multi-media Terminal Framework provides VoIP application developers with an extensible framework for developing IP-based videophones, IP phones and IADs. It shields the application from the myriad tasks that a terminal must implement. It also enables application developers to concentrate on value-added services rather than on the intricacies of VoIP signaling, security, call state handling, and standard supplementary services. A complete terminal solution can be achieved by integrating the RADVISION Multi-media Terminal Framework components with third-party DSP services and telephony hardware. For more information, see the *RADVISION Multi-media Terminal Framework Programmer Guide*.

MESSAGE MODULE

The Message API provides access to SIP message content enabling the application to browse and edit any part of the message. There are API functions for complete SIP messages and also for message parts such as headers and URLs.

The Message API is also the interface to the parsers and encoders provided with the SIP Stack. You can copy, store and re-use messages and message-parts.

This part includes the following sections:

- Message Object Functions
- Address Functions
- Allow Header Functions
- Authentication Header Functions
- Authorization Header Functions
- Contact Header Functions
- Content-Disposition Header Functions
- Content-Type Header Functions
- CSeq Header Functions
- Date Header Functions
- Expires Header Functions
- Other Header Functions
- Party Header Functions
- RAck Header Functions
- Referred-By Header Functions

- Refer-To Header Functions
- Retry-After Header Functions
- Route-Hop Header Functions
- RSeq Header Functions
- Via Header Functions
- Functions for All Headers
- Body Functions
- Body Part Functions

1

MESSAGE OBJECT FUNCTIONS

A message object (*message*) represents a SIP message and holds all SIP message information including start-line, headers and message body. Message object functions enable you to construct, destruct, copy, encode, parse, access and change message object parameters and manage the list of headers in *messages*. This section contains message object functions found in the *RvSipMsg.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipMsgConstruct()
- RvSipMsgDestruct()
- RvSipMsgCopy()
- RvSipMsgEncode()
- RvSipMsgParse()
- RvSipMsgConstructHeaderInMsgByType()
- RvSipMsgForceCompactForm()

RvSipMsgConstruct()

DESCRIPTION

Constructs a SIP *message*. The *message* is constructed on a page taken from a given memory pool.

SYNTAX

```
RvStatus RvSipMsgConstruct(
    IN  RvSipMsgMgrHandle    hMsgMgr,
    IN  HRPOOL               hPool,
    OUT RvSipMsgHandle*      hSipMsg);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool. The RvSipMsgConstruct() function uses a page from this pool for the newly created *message*.

hSipMsg

The handle of the new *message*.

RETURN VALUES

Returns RvStatus.

RvSipMsgDestruct()

DESCRIPTION

A destructor. Destroys the *message* and frees all the object resources.

SYNTAX

```
void RvSipMsgDestruct(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message* to destruct.

RETURN VALUES

None.

RvSipMsgCopy()

DESCRIPTION

Copies all parameters from a source *message* to a destination *message*. You should construct the destination *message* before using this function.

SYNTAX

```
RvStatus RvSipMsgCopy(
    INOUT RvSipMsgHandle    hDestination,
    IN    RvSipMsgHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination message to which values are copied.

hSource

The handle of the source *message*.

RETURN VALUES

Returns RvStatus.

RvSipMsgEncode()

DESCRIPTION

Encodes a *message* to a textual SIP message. The textual SIP message is placed on a page taken from a given memory pool. To copy the message from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipMsgEncode(
    IN  RvSipMsgHandle    hSipMsg,
    IN  HRPOOL            hPool,
    OUT HPAGE*            hPage,
    OUT RvUint32*         length);
```

PARAMETERS

hSipMsg

The handle of the message to encode.

hPool

The handle of a pool.

hPage

The memory page allocated to hold the encoded message.

length

The length of the encoded message.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipMsgParse()

DESCRIPTION

Parses a SIP textual message into a *message*. All the textual parameters are placed inside the object. Note that the given message buffer must be ended with two CRLFs. One CRLF must separate between headers in the message.

SYNTAX

```
RvStatus RvSipMsgParse(
    IN RvSipMsgHandle    hMsg,
    IN RvChar*           buffer,
    IN RvInt32           bufferLen);
```

PARAMETERS

hMsg

The handle of the *message*.

buffer

The buffer containing a textual message ending with two CRLFs.

bufferLen

The length of the buffer.

RETURN VALUES

Returns RvStatus.

RvSipMsgConstructHeaderInMsgByType()

DESCRIPTION

Constructs a header in a message according to the header type.

SYNTAX

```
RvStatus RvSipMsgConstructHeaderInMsgByType(
    IN  RvSipMsgHandle      hMsg,
    IN  RvSipHeaderType     eType,
    IN  RvBool              bPushHeaderAtHead,
    OUT void**              phHeader);
```

PARAMETERS

hMsg

The handle of the message.

eType

The type of the header to construct.

bpushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE), or to the tail (RV_FALSE).

phHeader

The handle of the header structure.

RETURN VALUES

Returns RvStatus.

RvSipMsgForceCompactForm()

DESCRIPTION

Instructs a *message* to force all its headers to use compact form. If compact form is forced and a message is encoded, all headers that can take compact form will be encoded with compact form. Headers that are handled by the Stack as Other headers and are added to the message by the application will not use compact form. It is the responsibility of the application to set such header names with compact form. Other headers that are added to the message by the Stack, such as supported headers, will be encoded with compact form.

SYNTAX

```
RvStatus RvSipMsgForceCompactForm(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            bForceCompactForm)
```

PARAMETERS

hSipMsg

The handle of the *message*.

bForceCompactForm

Specifies whether or not to force compact form.

REMARKS

- This flag effects only the encoding results of full messages. If you retrieve a specific header from the message and encode it, it will not be encoded with compact form if this specific header is not marked with compact form.
- The address header list and headers in a multipart mime bodies are not effected by this flag and will not be encoded with compact form.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipMsgGetStringLength()
- RvSipMsgGetMsgType()
- RvSipMsgGetRequestMethod()
- RvSipMsgGetStrRequestMethod()
- RvSipMsgSetMethodInRequestLine()
- RvSipMsgGetRequestUri()
- RvSipMsgSetRequestUri()
- RvSipMsgGetStatusCode()
- RvSipMsgSetStatusCode()
- RvSipMsgGetReasonPhrase()
- RvSipMsgSetReasonPhrase()
- RvSipMsgGetHeader()
- RvSipMsgGetHeaderByType()
- RvSipMsgGetHeaderByName()
- RvSipMsgGetHeaderExt()
- RvSipMsgGetHeaderByTypeExt()
- RvSipMsgGetHeaderByNameExt()
- RvSipMsgGetBadSyntaxHeaders()
- RvSipMsgPushHeader()
- RvSipMsgRemoveHeaderAt()
- RvSipMsgGetToHeader()
- RvSipMsgSetToHeader()
- RvSipMsgGetFromHeader()
- RvSipMsgSetFromHeader()
- RvSipMsgGetContentLengthHeader()
- RvSipMsgSetContentLengthHeader()
- RvSipMsgGetCSeqHeader()
- RvSipMsgSetCSeqHeader()
- RvSipMsgGetCallIdHeader()
- RvSipMsgSetCallIdHeader()
- RvSipMsgGetContentTypeHeader()

- RvSipMsgSetContentTypeHeader()
- RvSipMsgGetBody()
- RvSipMsgSetBody()
- RvSipMsgGetBodyObject()
- RvSipMsgSetBodyObject()
- RvSipMsgGetRpoolString()
- RvSipMsgSetRpoolString()
- RvSipMsgGetStrBadSyntaxStartLine()
- RvSipMsgStartLineFix()
- RvSipMsgSetCallIdCompactForm()
- RvSipMsgGetCallIdCompactForm()
- RvSipMsgSetContentLengthCompactForm()
- RvSipMsgGetContentLengthCompactForm()

RvSipMsgGetStringLength()

DESCRIPTION

Some of the message parameters are kept in a string format, for example, the reason phrase of a response message. In order to get such a parameter from the *message*, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

Note If the message body is of the multipart type, it may be represented as a list of body parts, and not as a string. To retrieve the body string length, the body object must be encoded if it is presented as a list of body parts. This function will encode the message body on a temporary memory page whenever the massage body object contains a list of body parts (to calculate the message body string length). If this function did not succeed to encode the message body (an error occurred while encoding), 0 is returned and an appropriate ERROR can be found in the log.

SYNTAX

```
RvUint RvSipMsgGetStringLength(
    IN RvSipMsgHandle            hSipMsg,
    IN RvSipMessageStringName    stringName);
```

PARAMETERS

hHeader

The handle of the *message*.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipMsgGetMsgType()

DESCRIPTION

Gets the message type: request, response or undefined.

SYNTAX

```
RvSipMsgType RvSipMsgGetMsgType(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message*.

RETURN VALUES

Returns the enumeration of the message type. For more information, see Message Type Definitions.

RvSipMsgGetRequestMethod()

DESCRIPTION

Gets the method type enumeration from the *message* request line. For more information, see RvSipMethodType in Message Type Definitions.

SYNTAX

```
RvSipMethodType RvSipMsgGetRequestMethod(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message*.

RETURN VALUES

Returns the request method type.

REMARKS

If this function returns RVSIP_METHOD_OTHER, call RvSipMsgGetStrRequestMethod() to get the method in a string format.

RvSipMsgGetStrRequestMethod()

DESCRIPTION

Copies the method type string from the *message* request line into a given buffer. Use this function if RvSipMsgGetRequestMethod() returns RVSIP_METHOD_OTHER. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipMsgGetStrRequestMethod(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetMethodInRequestLine()

DESCRIPTION

Sets the method type in the request line of the *message*. The function gets an enumeration and string as input. The string is set as method only if the enumeration is RVSIP_METHOD_OTHER. Otherwise, the string is ignored.

SYNTAX

```
RvStatus RvSipMsgSetMethodInRequestLine(
    INOUT RvSipMsgHandle     hSipMsg,
    IN    RvSipMethodType    eRequestMethod,
    IN    RvChar*            strMethod);
```

PARAMETERS

hSipMsg

The handle of a *message*.

eRequestMethod

Request method type to set in the message.

strMethod

A textual string which indicates the method type when eRequsetMethod is RVSIP_METHOD_OTHER.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetRequestUri()

DESCRIPTION

Gets the request URI address object handle.

SYNTAX

```
RvSipAddressHandle RvSipMsgGetRequestUri(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of a *message*.

RETURN VALUES

Returns a handle of the URI address object or NULL if no URI address exists.

RvSipMsgSetRequestUri()

DESCRIPTION

Sets the request URI address in the *message* request line.

SYNTAX

```
RvStatus RvSipMsgSetRequestUri(
    IN RvSipMsgHandle        hSipMsg,
    IN RvSipAddressHandle    hSipUrl);
```

PARAMETERS

hSipMsg

The handle of a *message*.

hSipUrl

The handle of an address object with the request URI.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetStatusCode()

DESCRIPTION

Gets the response code from the *message* start-line.

SYNTAX

```
RvSipMsgGetStatusCode(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message*.

RETURN VALUES

Returns a response code or UNDEFINED if there is no response code or the *message* represents a request message.

RvSipMsgSetStatusCode()

DESCRIPTION

Sets a status code in the *message* status line. This will set the *message* to represent a response message.

SYNTAX

```
RvStatus RvSipMsgSetStatusCode(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            code,
    IN RvBool            insertDefaultReasonPhrase);
```

PARAMETERS

hSipMsg

The handle of a *message*.

code

Status response code to be set in the *message*.

insertDefaultReasonPhrase

Determines whether or not to insert a default reason phrase into the status line. the reason phrase is set according to the code number.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetReasonPhrase()

DESCRIPTION

Copies the reason phrase from the *message* status line into a given buffer. If the *bufferLen* size adequate, the function copies the parameter into the *strBuffer*. Otherwise, it returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipMsgGetReasonPhrase(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetReasonPhrase()

DESCRIPTION

Sets the reason phrase in the *message* status line.

SYNTAX

```
RvStatus RvSipMsgSetReasonPhrase(
    IN RvSipMsgHandle    hSipMsg,
    IN RvChar*           strReasonPhrase);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strReasonPhrase

The reason phrase to be set in the *message*. If NULL is supplied, the existing reason phrase of the message is removed

RETURN VALUES

Returns RvStatus.

RvSipMsgGetHeader()

DESCRIPTION

Gets a header from the header list. The *message* holds most headers in a sequential list except, To, From, CallId, Cseq, ContentLength and Content-Type headers. The function returns the handle of the requested header as a void pointer. You should check the pHeaderType parameter and cast the return value to the appropriate header handle.

Note The RvSipMsgGetHeaderExt() function is an extension of this function.

SYNTAX

```
void* RvSipMsgGetHeader(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvSipHeadersLocation          location,
    INOUT RvSipHeaderListElemHandle*    hListElem,
    OUT   RvSipHeaderType*              pHeaderType);
```

PARAMETERS

hSipMsg

The handle of the *message*.

location

The location on list: next, previous, first or last.

hListElem

The handle of the current position in the list. Supply this value if you chose next or previous in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

pHeaderType

The type of the retrieved header.

RETURN VALUES

Returns the header handle as void*, or NULL if there is no header to retrieve.

RvSipMsgGetHeaderByType()

DESCRIPTION

Gets a header by type from the header list. The *message* holds most headers in a sequential list except, To, From, CallId, Cseq, ContentLength and Content-Type headers. The function returns the handle of the requested header as a void pointer. You should cast the return value to the header type you requested.

Note The RvSipMsgGetHeaderByTypeExt() function is an extension of this function.

SYNTAX

```
void* RvSipMsgGetHeaderByType(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvSipHeaderType               eHeaderType,
    IN    RvSipHeadersLocation          location,
    INOUT RvSipHeaderListElemHandle*    hListElem);
```

PARAMETERS

hSipMsg

The handle of the *message*.

eHeaderType

The header type to be retrieved.

location

The location in the header list: next, previous, first or last.

hListElem

The handle of the current position in the list. Supply this value if you chose next or previous in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

RETURN VALUES

Returns the header handle as void*, or NULL if there is no header to retrieve.

RvSipMsgGetHeaderByName()

DESCRIPTION

Gets a header by name from the header list. The *message* holds most headers in a sequential list except, To, From, CallId, Cseq, ContentLength and Content-Type headers. This function should be used only for headers of type RVSIP_HEADERTYPE_OTHER.

Note The RvSipMsgGetHeaderByNameExt() function is an extension of this function.

SYNTAX

```
RvSipOtherHeaderHandle RvSipMsgGetHeaderByName(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvChar*                       strName,
    IN    RvSipHeadersLocation          location,
    INOUT RvSipHeaderListElemHandle*    hListElem);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strName

The header name to be retrieved.

location

The location in the header list: next, previous, first or last.

hListElem

The handle of the current position in the list. Supply this value if you chose next or previous in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

RETURN VALUES

Returns the Other header handle, or NULL if no Other header handle with the same name exists.

RvSipMsgGetHeaderExt()

DESCRIPTION

Gets a header from the header list. The *message* holds most headers in a sequential list except for the To, From, CallId, Cseq, ContentLength and Content-Type headers. This function returns the handle of the requested header as a void pointer.

You should check the *pHeaderType* parameter and cast the return value to the appropriate header handle. The header list can hold both valid syntax and bad-syntax headers. Using the *eOption* parameter, you should specify whether you wish to get any type of header or only headers with valid syntax.

Note This function extends the functionality of the RvSipMsgGetHeader() function.

SYNTAX

```
void* RvSipMsgGetHeaderExt(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvSipHeadersLocation          location,
    IN    RvSipMsgHeadersOption         eOption,
    INOUT RvSipHeaderListElemHandle*    hListElem,
    OUT   RvSipHeaderType*              pHeaderType);
```

PARAMETERS

hSipMsg

The handle of the *message*.

location

The location on list: next, previous, first or last.

eOption

Specifies whether the application wishes to get only legal syntax headers or any header (legal and illegal headers).

hListElem

The handle of the current position in the list. Supply this value if you chose next or previous in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

pHeaderType

The type of retrieved header.

RETURN VALUES

Returns the header handle as void*, or NULL if there is no header to retrieve.

RvSipMsgGetHeaderByTypeExt()

DESCRIPTION

Gets a header by type from the header list. The *message* holds most headers in a sequential list except for the To, From, CallId, Cseq, ContentLength and Content-Type headers. This function returns the handle of the requested header as a void pointer.

You should cast the return value to the header type you requested. The header list can hold both valid syntax and bad-syntax headers. Using the *eOption* parameter, you should specify whether you wish to get any type of header or only headers with valid syntax.

Note This function extends the functionality of RvSipMsgGetHeaderByType function.

SYNTAX

```
void* RvSipMsgGetHeaderByTypeExt(
    IN RvSipMsgHandle                   hSipMsg,
    IN RvSipHeaderType                  eHeaderType,
    IN RvSipHeadersLocation             location,
    IN RvSipMsgHeadersOption            eOption,
    INOUT RvSipHeaderListElemHandle*    hListElem);
```

PARAMETERS

hSipMsg

The handle of the *message*.

eHeaderType

The header type to be retrieved.

location

The location on the list: next, previous, first or last.

eOption

Specifies whether the application wishes to get only legal syntax headers, or any header (legal and illegal headers).

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

RETURN VALUES

Returns the header handle as void*, or NULL if there is no header to retrieve.

RvSipMsgGetHeaderByNameExt()

DESCRIPTION

Gets a header by name from the header list. The *message* holds most headers in a sequential list except for the To, From, CallId, Cseq, ContentLength and Content-Type headers. This function should be used only for headers of the RVSIP_HEADERTYPE_OTHER type. The header list can hold both valid syntax and bad-syntax headers. Using the *eOption* parameter, you should specify whether you wish to get any type of header or only headers with valid syntax.

Note This function extends the functionality of RvSipMsgGetHeaderByName function.

STANDS

```
RvSipOtherHeaderHandle RvSipMsgGetHeaderByNameExt(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvChar*                       strName,
    IN    RvSipHeadersLocation          location,
    IN    RvSipMsgHeadersOption         eOption,
    INOUT RvSipHeaderListElemHandle*    hListElem);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strName

The header name to be retrieved.

location

The location on the list: next, previous, first or last.

eOption

Specifies whether the application wishes to get only legal syntax headers, or any header (legal and illegal headers).

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

RETURN VALUES

Returns the Other header handle, or NULL if an Other header handle with the same name does not exist.

RvSipMsgGetBadSyntaxHeaders()

DESCRIPTION

Gets a header with a syntax error from the message. The *message* holds most headers in a sequential list. To, From, CallId, Cseq, ContentLength and Content-Type headers are held separately. This function scans all headers in the *message* (the ones that are in the header list and the ones that are not) and retrieves only headers with syntax errors.

This function treats all headers as if they were located in one virtual list. The virtual list includes the headers that are in the header list and the headers that are not. This function returns the handle of the requested header as a void pointer. You should check the *pHeaderType* parameter and cast the return value to the appropriate header handle.

Note This function does not check the Content-Length header value. To check the validity of the Content-Length header, use RvSipMsgGetContentLengthHeader().

SYNTAX

```
void* RvSipMsgGetBadSyntaxHeaders(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvSipHeadersLocation          location,
    INOUT RvSipHeaderListElemHandle*    hListElem,
    OUT   RvSipHeaderType*              pHeaderType);
```

PARAMETERS

hSipMsg

The handle of the *message*.

location

The location on the list: next, previous, first or last.

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to the requested header in the list.

hListElem

The handle of the current position in the list. Supply this value if you chose “next” or “previous” in the location parameter. This is also an output parameter and will be set with a link to the requested header in the list.

pHeaderType

The type of retrieved header.

RETURN VALUES

Returns the header handle as void*, or NULL if there is no header to retrieve.

RvSipMsgPushHeader()

DESCRIPTION

Inserts a given header into the header list based on a given location. For example, first, last, before or after a given element. The *message* holds most headers in a sequential list except, To, From, CallId, Cseq, ContentLength and Content-Type headers.

The header you supply is copied before it is inserted into the list. The *pNewHeader* output parameter contains the handle of the actual header pushed into the list. You should use this handle of refer to the header pushed into the list.

SYNTAX

```
RvStatus RvSipMsgPushHeader(
    IN    RvSipMsgHandle                hSipMsg,
    IN    RvSipHeadersLocation          location,
    IN    void*                         pHeader,
    IN    RvSipHeaderType               eHeaderType,
    INOUT RvSipHeaderListElemHandle*    hListElem,
    OUT   void**                        pNewHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

location

The location in the header list: next, previous, first or last.

pHeader

The handle of the header pushed into the list.

eHeaderType

Type of the header to be pushed.

hListElem

The handle of the current position in the list. Supply this value if you chose next or previous in the location parameter. This is also an output parameter and will be set with a link to requested header in the list.

pNewHeader

The handle of the actual header that is inserted into the list.

RETURN VALUES

Returns RvStatus.

RvSipMsgRemoveHeaderAt()

DESCRIPTION

Removes an header from the header list. The *message* holds most headers in a sequential list except, To, From, CallId, Cseq, ContentLength and Content-Type headers. You should supply this function with the list element of the header you wish to remove.

SYNTAX

```
RvStatus RvSipMsgRemoveHeaderAt(
    IN RvSipMsgHandle               hSipMsg,
    IN RvSipHeaderListElemHandle    hListElem);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hListElem

The handle of the list element to be removed.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetToHeader()

DESCRIPTION

Gets the handle of the To header from the *message*.

SYNTAX

```
RvSipPartyHeaderHandle RvSipMsgGetToHeader(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of a *message*.

RETURN VALUES

Returns RvSipPartyHeaderHandle, or NULL if the message does not contain a To header.

RvSipMsgSetToHeader()

DESCRIPTION

Sets a To header in the *message*. If there is no To header in the message, the function constructs a To header.

SYNTAX

```
RvStatus RvSipMsgSetToHeader(
    IN RvSipMsgHandle            hSipMsg,
    IN RvSipPartyHeaderHandle    hToHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hToHeader

The handle of the To header to be set. If a NULL is supplied, the existing To header is removed from the message.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetFromHeader()

DESCRIPTION

Gets a handle of the From header from the *message*.

SYNTAX

```
RvSipPartyHeaderHandle RvSipMsgGetFromHeader(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of a *message*.

RETURN VALUES

MsgPartyHeader handle, or NULL if the message does not contain a From header.

RvSipMsgSetFromHeader()

DESCRIPTION

Sets a From header in the *message*. If no From header exists in the message, the function constructs a From header.

SYNTAX

```
RvStatus RvSipMsgSetFromHeader(
    IN RvSipMsgHandle            hSipMsg,
    IN RvSipPartyHeaderHandle    hFromHeader);
```

PARAMETERS

hSipMsg

The handle of a *message*.

hFromHeader

The handle of the From header to be set. If a NULL value is supplied, the existing From header is removed from the message.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetContentLengthHeader()

DESCRIPTION

Gets the content length of the *message*. The content length indicates the size of the body string.

SYNTAX

```
RvInt32 RvSipMsgGetContentLengthHeader(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message*.

RETURN VALUES

Returns content length or UNDEFINED if no body is set in the *message*.

REMARKS

- If the body is of type multipart and holds a list of body parts, this function will return UNDEFINED.
- To receive the content-length of a multipart body use RvSipMsgGetStringLength() with body enumeration.

RvSipMsgSetContentLengthHeader()

DESCRIPTION

Sets a Content Length value in the *message*. The Content Length indicates the size of the message body.

SYNTAX

```
RvStatus RvSipMsgSetContentLengthHeader(
    IN RvSipMsgHandle    hSipMsg,
    IN RvInt32           contentLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

contentLen

The Content Length value to be set.

RETURN VALUES

Returns RvStatus.

REMARKS

If the body is of type multipart and holds a list of body part objects, this function will return RV_ERROR_UNKNOWN. In this case, the content length is determine by the size of the encoded body part list.

RvSipMsgGetCSeqHeader()

DESCRIPTION

Gets the handle of the Cseq header from the *message*.

SYNTAX

```
RvSipCSeqHeaderHandle RvSipMsgGetCSeqHeader(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of a *message*.

RETURN VALUES

Returns the CSeq header handle, or NULL if no CSeq header handle exists.

RvSipMsgSetCSeqHeader()

DESCRIPTION

Sets a CSeq header in the *message*. If there is no CSeq header in the message, the function constructs one.

SYNTAX

```
RvStatus RvSipMsgSetCSeqHeader(
    IN RvSipMsgHandle           hSipMsg,
    IN RvSipCSeqHeaderHandle    hCSeqHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hCSeqHeader

The handle of the Cseq header to be set. If NULL value is supplied, the existing CSeq header is removed from the message.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetCallIdHeader()

DESCRIPTION

Copies the *message* Call-ID into a given buffer. If the *bufferLen* size is adequate, the function copies the Call-ID into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipMsgGetCallIdHeader(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetCallIdHeader()

DESCRIPTION

Sets a Call-ID in the *message*.

SYNTAX

```
RvStatus RvSipMsgSetCallIdHeader(
    IN RvSipMsgHandle    hSipMsg,
    IN RvChar*           strCallId);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strCallId

The new Call-ID string to be set in the *message*. If a NULL value is supplied, the existing Call-ID is removed from the message.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetContentTypeHeader()

DESCRIPTION

Copies the Content-Type of the *message* into a given buffer. If the *bufferLen* size is adequate, the function copies the Content-Type, into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipMsgGetContentTypeHeader(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetContentTypeHeader()

DESCRIPTION

Sets a Content-Type string in the *message*.

SYNTAX

```
RvStatus RvSipMsgSetContentTypeHeader(
    IN RvSipMsgHandle    hSipMsg,
    IN RvChar*           pContentType);
```

PARAMETERS

hSipMsg

The handle of a *message*.

pContentType

The ContentType string to be set in the *message*. If a NULL value is supplied, the existing Content-Type header is removed from the message.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetBody()

DESCRIPTION

Copies the body from the *message* into a given buffer. If the *bufferLen* is adequate, the function copies the message body into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipMsgGetBody(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

REMARKS

If the message body is of multipart type, it may be represented as a list of body parts, and not as a string. In this case, the body object will first be encoded and then will be copied to the given buffer.

RvSipMsgSetBody()

DESCRIPTION

Sets a body in the *message* and updates the content-length parameter with the body length. Remember to update the Content-Type value according to the new body type.

SYNTAX

```
RvStatus RvSipMsgSetBody(
    IN RvSipMsgHandle    hSipMsg,
    IN RvChar*           strBody);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBody

SIP message body to be set in the *message*. If NULL is supplied, the existing body is removed and the *message* contentLength is set to zero.

RETURN VALUES

Returns RvStatus.

REMARKS

You can use RvSipMsgSetBody() **only** when the body string does not contain NULL characters.

- To set body string that contains NULL characters you must first get the body object from the message using RvSipMsgGetBodyObject(). Then you can use RvSipBodySetBodyStr() to set the body string.
- You can also use RvSipBodyConstructInMsg() to construct a new body object in the *message*.

RvSipMsgGetBodyObject()

DESCRIPTION

Gets the body object from the given *message*. The handle of the body object is returned. Any changes that are made to this body object will be reflected in the message.

SYNTAX

```
RvSipBodyHandle RvSipMsgGetBodyObject(
    IN RvSipMsgHandle    hSipMsg);
```

PARAMETERS

hSipMsg

The handle of the *message*.

RETURN VALUES

Returns the handle of the body object.

RvSipMsgSetBodyObject()

DESCRIPTION

Sets a body object in a given *message*. The given body is copied to the message. If the body object contains a Content-Type header, it is copied to the *message* as well.

SYNTAX

```
RvStatus RvSipMsgSetBodyObject(
    IN RvSipMsgHandle     hSipMsg,
    IN RvSipBodyHandle    hBody);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hBody

The body object to be set to the *message*. If NULL is supplied, the existing body is removed from the *message*.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetRpoolString()

DESCRIPTION

Copies a string parameter from the message into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipMsgGetRpoolString(
    IN    RvSipMsgHandle            hSipMsg,
    IN    RvSipMessageStringName    eStringName,
    INOUT RPOOL_Ptr                 *pRpoolPtr);
```

PARAMETERS

hSipMsg

The handle of the *message*.

eStringName

The string that the user wishes to get. In the current version, RVSIP_MSG_CALL_ID is the only supported string.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function sets the returned offset to UNDEFINED, this means that the parameter was not set to the Other header.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the *message*. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipMsgSetRpoolString(
    IN RvSipMsgHandle            hSipMsg,
    IN RvSipMessageStringName    eStringName,
    IN RPOOL_Ptr                 *pRpoolPtr);
```

PARAMETERS

hSipMsg

The handle of the *message*.

eStringName

The string the user wishes to set. In the current version, RVSIP_MSG_CALL_ID is the only supported string.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetStrBadSyntaxStartLine()

DESCRIPTION

Copies the bad-syntax start line from the *message* into a given buffer. When a message is received and the start line contains a syntax error, the start line is kept as a separate bad-syntax string. This function retrieves this string. If the *bufferLen* size adequate, this function copies the parameter into the *strBuffer*. Otherwise, it returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length. Use this function in RvSipTransportBadSyntaxStartLineMsgEv() event implementation to check and fix the defected start-line.

Note If the start line is valid the function will return RV_ERROR_NOT_FOUND and the *actualLen* parameter will be set to 0.

SYNTAX

```
RvStatus RvSipMsgGetStrBadSyntaxStartLine(
    IN  RvSipMsgHandle    hSipMsg,
    IN  RvChar*           strBuffer,
    IN  RvUint            bufferLen,
    OUT RvUint*           actualLen);
```

PARAMETERS

hSipMsg

The handle of the *message*.

strBuffer

The buffer with which to fill the bad-syntax start line.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax start line + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMsgStartLineFix()

DESCRIPTION

Fixes a start-line with bad-syntax information. When a message is received with a syntax error in the start line, the start line string is kept as a separate bad-syntax string. You use this function to fix the start line by parsing a given correct start line string to the supplied *message*. If parsing succeeds, the function places all parameters inside the message start line and removes the bad-syntax string. If parsing fails, the bad-syntax string remains untouched. Use this function in the RvSipTransportBadSyntaxStartLineMsgEv() event implementation to fix a defected start-line.

SYNTAX

```
RvStatus RvSipMsgStartLineFix(
    IN RvSipMsgHandle    hSipMsg,
    IN RvChar*           pFixedBuffer);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pFixedBuffer

The buffer containing a legal start-line value.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetCallIdCompactForm()

DESCRIPTION

Instructs the message to use the compact header name when the Call-ID header is encoded.

SYNTAX

```
RvStatus RvSipMsgSetCallIdCompactForm(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            bIsCompact);
```

PARAMETERS

hSipMsg

The handle of the *message*.

bIsCompact

Specifies whether or not to use a compact form.

RETURN VALUES

Returns RvStatus.

RvSipMsgGetCallIdCompactForm()

DESCRIPTION

Gets the compact form indication of the Call-ID header found in the supplied message.

SYNTAX

```
RvStatus RvSipMsgGetCallIdCompactForm(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            *pbIsCompact)
```

PARAMETERS

hSipMsg

The handle of the *message*.

bIsCompact

Specifies whether or not compact form is used. RV_TRUE if the header uses compact form, otherwise RV_FALSE.

RETURN VALUES

Returns RvStatus.

RvSipMsgSetContentLengthCompactForm()

DESCRIPTION

Instructs the message to use the compact header name when the Content-Length header is encoded.

SYNTAX

```
RvStatus RvSipMsgSetContentLengthCompactForm(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            bIsCompact);
```

PARAMETERS

hSipMsg

The handle of the *message*.

bIsCompact

Specifies whether or not to use a compact form.

RETURN VALUE

Returns RvStatus.

RvSipMsgGetContentLengthCompactForm()

DESCRIPTION

Gets the compact form indication of the Content-Length header found in the supplied message.

SYNTAX

```
RvStatus RvSipMsgGetContentLengthCompactForm(
    IN RvSipMsgHandle    hSipMsg,
    IN RvBool            *pbIsCompact)
```

PARAMETERS

hSipMsg

The handle of the *message*.

pbIsCompact

Specifies whether or not compact form is used. RV_TRUE if the header uses compact form, otherwise RV_FALSE.

RETURN VALUE

Returns RvStatus.

Get and Set Functions
RvSipMsgGetContentLengthCompactForm()

2

ADDRESS FUNCTIONS

The Address API functions enable you to construct, copy, encode, parse, access and change address object (*address*) parameters. The Address API includes dedicated functions for both SIP URL and Absolute URI. This section contains Address functions found in the *RvSipAddress.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions
- Address Comparison Functions
- SIP URL Get and Set Functions
- Absolute URI Get and Set Functions
- TEL URI Get and Set Functions
- Diameter Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAddrConstructInStartLine()
- RvSipAddrConstructInPartyHeader()
- RvSipAddrConstructInContactHeader()
- RvSipAddrConstructInAuthorizationHeader()
- RvSipAddrConstructInRouteHopHeader()
- RvSipAddrConstructInReferToHeader()
- RvSipAddrConstructInReferredByHeader()
- RvSipAddrConstruct()
- RvSipAddrCopy()
- RvSipAddrEncode()
- RvSipAddrParse()
- RvSipAddrConstructInContentTypeHeader()
- RvSipAddrConstructInContentIDHeader()
- RvSipAddrConstructAndParse()
- RvSipAddrConstructInPProfileKeyHeader()
- RvSipAddrConstructInPUserDatabaseHeader()
- RvSipAddrDiameterUriIsSecure()

RvSipAddrConstructInStartLine()

DESCRIPTION

Constructs an *address* inside the Request line of a given message. The new *address* is set as the request URI of the Request line. The handle of the *address* is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInStartLine(
    IN  RvSipMsgHandle          hSipMsg,
    IN  RvSipAddressType        eAddrType,
    OUT RvSipAddressHandle*     hSipAddr);
```

PARAMETERS

hSipMsg

The handle of the SIP message.

eAddrType

The type of *address* to construct. In this version, only a URL address.

hSipAddr

The handle of the newly created *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInPartyHeader()

DESCRIPTION

Constructs the address-spec *address* inside a given To or From Party header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInPartyHeader(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  RvSipAddressType          eAddrType,
    OUT RvSipAddressHandle*       hSipAddr);
```

PARAMETERS

hHeader

The handle of the Party header.

eAddrType

The type of *address* to construct—in this version, only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInContactHeader()

DESCRIPTION

Constructs the address-spec *address* inside a given Contact header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInContactHeader(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvSipAddressType            eAddrType,
    OUT RvSipAddressHandle*         hSipAddr);
```

PARAMETERS

hHeader

The handle of the Contact header.

eAddrType

The type of *address* to construct—in this version, only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInAuthorizationHeader()

DESCRIPTION

Constructs the digest URI *address* inside a given Authorization header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInAuthorizationHeader(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvSipAddressType                  eAddrType,
    OUT RvSipAddressHandle*               hSipAddr);
```

PARAMETERS

hHeader

The handle of the Authorization header.

eAddrType

The type of *address* to construct—in this version, only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInRouteHopHeader()

DESCRIPTION

Constructs the address-spec *address* inside a given Route-Hop header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInRouteHopHeader(
    IN  RvSipRouteHopHeaderHandle    hHeader,
    IN  RvSipAddressType             eAddrType,
    OUT RvSipAddressHandle*          hSipAddr);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

eAddrType

The type of *address* to construct. In the current version, *eAddrType* is only a URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInReferToHeader()

DESCRIPTION

Constructs the address-spec *address* inside a given Refer-To header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInReferToHeader(
    IN  RvSipReferToHeaderHandle    hHeader,
    IN  RvSipAddressType            eAddrType,
    OUT RvSipAddressHandle*         hSipAddr);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

eAddrType

The type of *address* to construct. In the current version, *eAddrType* is only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInReferredByHeader()

DESCRIPTION

Constructs the address-spec *address* inside a given Referred-By header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInReferredByHeader(
    IN  RvSipReferredByHeaderHandle    hHeader,
    IN  RvSipAddressType               eAddrType,
    OUT RvSipAddressHandle*            hSipAddr);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

eAddrType

The type of *address* to construct. In the current version, *eAddrType* is only a URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstruct()

DESCRIPTION

Constructs and initializes a stand-alone *address*. The object is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAddrConstruct(
    IN  RvSipMsgMgrHandle      hMsgMgr,
    IN  HRPOOL                 hPool,
    IN  HPAGE                  hPage,
    IN  RvSipAddressType       eAddrType,
    OUT RvSipAddressHandle*    hSipAddr);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object uses.

hPage

The handle of the memory page that the object uses.

eAddrType

The type of *address* URL to construct.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrCopy()

DESCRIPTION

Copies all parameters from a source *address* to a destination *address*. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAddrCopy(
    INOUT RvSipAddressHandle    hDestination,
    IN    RvSipAddressHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination *address*.

hSource

The handle of the source *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrEncode()

DESCRIPTION

Encodes an *address* to a textual *address*. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAddrEncode(
    IN  RvSipAddressHandle    hSipAddr,
    IN  HRPOOL                hPool,
    OUT HPAGE*                hPage,
    OUT RvUint32*             length);
```

PARAMETERS

hSipAddr

The handle of the *address*.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded object.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

Your application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipAddrParse()

DESCRIPTION

Parses a SIP textual *address*, such as, “sip:bob@radvision.com”, into an *address*. All the textual parameters are placed inside the object.

SYNTAX

```
RvStatus RvSipAddrParse(
    INOUT RvSipAddressHandle    hSipAddr,
    IN    RvChar*               buffer);
```

PARAMETERS

hSipAddr

A handle of the *address*.

buffer

The buffer containing a textual *address*.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct an *address* before using this function.

RvSipAddrConstructInContentTypeHeader()

DESCRIPTION

Constructs an *address* inside a given Content-Type. The new *address* will be set as a parameter within the Content-Type header. The handle of the *address* is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInContentTypeHeader(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvSipAddressType                eAddrType,
    OUT RvSipAddressHandle              *hSipAddr);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

eAddrType

The type of *address* to construct. In the current version, only URL.

hSipAddr

The handle of the newly created *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInContentIDHeader()

DESCRIPTION

Constructs an *address* inside a given Content-ID. The new *address* will be set as a parameter within the Content-ID header. The handle of the *address* is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInContentTypeHeader(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvSipAddressType                eAddrType,
    OUT RvSipAddressHandle              *hSipAddr);
```

PARAMETERS

hHeader

Thee handle of the Content-ID header.

eAddrType

The type of *address* to construct. In the present version, only URL.

hSipAddr

The handle of the newly created *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructAndParse()

DESCRIPTION

Constructs and parses a stand-alone *address*. The *address* is constructed on a given page taken from a specified pool. The type of the constructed *address* and the value of its fields are determined by parsing the given buffer. The handle of the newly allocated *address* is returned.

SYNTAX

```
RvStatus RvSipAddrConstructAndParse(
    IN  RvSipMsgMgrHandle    hMsgMgr,
    IN  HRPOOL               hPool,
    IN  HPAGE                hPage,
    IN  RvChar*              buffer,
    OUT RvSipAddressHandle*  hSipAddr);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object uses

hPage

The handle of the memory page that the object uses.

buffer

The buffer containing a textual *address*.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInPProfileKeyHeader()

DESCRIPTION

Constructs the address-spec *address* object inside a given PProfileKey header. The *address* handle is returned.

SYNTAX

```
RvStatus RvSipAddrConstructInPProfileKeyHeader(
    IN  RvSipPProfileKeyHeaderHandle    hHeader,
    IN  RvSipAddressType                eAddrType,
    OUT RvSipAddressHandle*             hSipAddr);
```

PARAMETERS

hHeader

The handle of the PProfileKey header.

eAddrType

The type of *address* to construct. In the current version, only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrConstructInPUserDatabaseHeader()

DESCRIPTION

Constructs the address-spec *address* object inside a given PUserDatabase header. The *address* handle is returned.

Note This *address* must be of type DiameterURI.

SYNTAX

```
RvStatus RvSipAddrConstructInPUserDatabaseHeader(
    IN  RvSipPUserDatabaseHeaderHandle    hHeader,
    IN  RvSipAddressType                  eAddrType,
    OUT RvSipAddressHandle*               hSipAddr);
```

PARAMETERS

hHeader

The handle of the PUserDatabase header.

eAddrType

The type of *address* to construct. In the current version, only URL.

hSipAddr

The handle of the newly constructed *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriIsSecure()

DESCRIPTION

Indicates whether or not the URL address scheme is secure (if the URL has "aaas://" or "aaa://" as a prefix).

SYNTAX

```
RvBool RvSipAddrDiameterUriIsSecure(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Is secure.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAddrGetStringLength()
- RvSipAddrGetAddrType()
- RvSipAddrGetRpoolString()
- RvSipAddrSetRpoolString()
- RvSipAddrGetAddressTypeFromString()

RvSipAddrGetStringLength()

DESCRIPTION

Some of the *address* parameters are kept in a string format, such as the *address* host name. In order to get such a parameter from the *address*, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAddrGetStringLength(
    IN RvSipAddressHandle        hSipAddr,
    IN RvSipAddressStringName    stringName);
```

PARAMETERS

hHeader

The handle of the *address*.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipAddrGetAddrType()

DESCRIPTION

Gets the *address* type of an *address*.

SYNTAX

```
RvSipAddressType RvSipAddrGetAddrType(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the *address*.

RETURN VALUES

Returns RvSipAddressType. For more information see RvSipAddressType in Message Type Definitions.

RvSipAddrGetRpoolString()

DESCRIPTION

Copies a string parameter from the *address* into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipAddrGetRpoolString(
    IN    RvSipAddressHandle        hAddr,
    IN    RvSipAddressStringName    eStringName,
    INOUT RPOOL_Ptr                 *pRpoolPtr);
```

PARAMETERS

hAddr

The handle of the *address*.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function sets the returned offset to UNDEFINED, this means that the parameter was not set to the Via header.

RETURN VALUES

Returns RvStatus.

RvSipAddrSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the *address*. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipAddrSetRpoolString(
    IN RvSipAddressHandle        hAddr,
    IN RvSipAddressStringName    eStringName,
    IN RPOOL_Ptr                 *pRpoolPtr);
```

PARAMETERS

hAddr

The handle of the *address*.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipAddrGetAddressTypeFromString()

DESCRIPTION

Gets the address type of an address in a string format.

SYNTAX

```
RvSipAddressType RvSipAddrGetAddressTypeFromString(
    IN RvSipMsgMgrHandle    hMsgMgr,
    IN RvChar               *strAddr);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

strAddr

The address in string format.

RETURN VALUES

Returns the type of the given address, or RVSIP_ADDRTYPE_UNDEFINED in case of failure.

ADDRESS COMPARISON FUNCTIONS

The Address Comparison functions are:

- RvSipAddrIsEqual()

RvSipAddrIsEqual()

DESCRIPTION

Compares two SIP *addresses*, as specified in RFC 2543.

SYNTAX

```
RvBool RvSipAddrIsEqual(
    IN RvSipAddressHandle    hAddress,
    IN RvSipAddressHandle    hOtherAddress);
```

PARAMETERS

hAddress

The handle of the first *address*.

hOtherAddress

The handle of the second *address* with which a comparison is being made.

RETURN VALUES

Returns RV_TRUE if the *addresses* are equal. Otherwise returns RV_FALSE.

SIP URL GET AND SET FUNCTIONS

The SIP URL Get and Set functions are:

- RvSipAddrUrlGetUser()
- RvSipAddrUrlSetUser()
- RvSipAddrUrlGetHost()
- RvSipAddrUrlSetHost()
- RvSipAddrUrlGetPortNum()
- RvSipAddrUrlSetPortNum()
- RvSipAddrUrlGetTtlNum()
- RvSipAddrUrlSetTtlNum()
- RvSipAddrUrlGetLrParam()
- RvSipAddrUrlSetLrParam()
- RvSipAddrUrlGetLrParamType()
- RvSipAddrUrlSetLrParamType()
- RvSipAddrUrlGetMethod()
- RvSipAddrUrlGetStrMethod()
- RvSipAddrUrlSetMethod()
- RvSipAddrUrlGetTransport()
- RvSipAddrUrlGetStrTransport()
- RvSipAddrUrlSetTransport()
- RvSipAddrUrlGetMaddrParam()
- RvSipAddrUrlSetMaddrParam()
- RvSipAddrUrlGetUserParam()
- RvSipAddrUrlGetStrUserParam()
- RvSipAddrUrlSetUserParam()
- RvSipAddrUrlGetHeaders()
- RvSipAddrUrlSetHeaders()
- RvSipAddrUrlGetHeadersList()
- RvSipAddrUrlSetHeadersList()
- RvSipAddrUrlGetCompParam()
- RvSipAddrUrlGetStrCompParam()
- RvSipAddrUrlSetCompParam()
- RvSipAddrUrlGetOtherParams()
- RvSipAddrUrlSetOtherParams()

- RvSipAddrUrlIsSecure()
- RvSipAddrUrlSetIsSecure()
- RvSipAddrUrlGetCPC()
- RvSipAddrUrlGetStrCPC()
- RvSipAddrUrlSetCPC()
- RvSipAddrUrlGetStrGr()
- RvSipAddrUrlGetGr()
- RvSipAddrUrlSetGr()
- RvSipAddrUrlSetSigCompIdParam()
- RvSipAddrUrlGetSigCompIdParam()

RvSipAddrUrlGetUser()

DESCRIPTION

Copies the user parameter from the *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetUser(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetUser()

DESCRIPTION

Sets the user string value in the *address* URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetUser(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strUser);
```

PARAMETERS

hSipAddr

The handle of the *address*.

strUser

The user value to be set in the object. If NULL is supplied, the existing strUser is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetHost()

DESCRIPTION

Copies the host parameter from the *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetHost(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetHost()

DESCRIPTION

Sets the host in the URL object or removes the existing host from the URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetHost(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strHost);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strHost

The host value to be set in the object. If NULL is supplied, the existing host is removed.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetPortNum()

DESCRIPTION

Gets the port number from the URL object.

SYNTAX

```
RvInt32 RvSipAddrUrlGetPortNum(
    IN RvSipAddressHandle    hSipUrl);
```

PARAMETERS

hSipUrl

The handle of the URL *address*.

RETURN VALUES

Returns the port number, or UNDEFINED if the port is not set.

RvSipAddrUrlSetPortNum()

DESCRIPTION

Sets the port number in the URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetPortNum(
    IN RvSipAddressHandle    hSipUrl,
    IN RvInt32               portNum);
```

PARAMETERS

hSipUrl

The handle of the URL *address*.

portNum

The port number value to be set in the object. In order to remove the port number from the *address*, you can set this parameter to UNDEFINED.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetTtlNum()

DESCRIPTION

Gets the ttl number from the URL object.

SYNTAX

```
RvSipAddrUrlTtlNum(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the ttl number, or UNDEFINED if ttl is not set.

RvSipAddrUrlSetTtlNum()

DESCRIPTION

Sets the ttl number in the *address* URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetTtlNum(
    IN RvSipAddressHandle    hSipAddr,
    IN RvInt16               ttlNum);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

ttlNum

The ttl number value to be set in the object. In order to remove the ttl number from the *address*, you can set this parameter to UNDEFINED.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetLrParam()

DESCRIPTION

Specifies whether or not the lr param exists in the address.

SYNTAX

```
RvBool RvSipAddrUrlGetLrParam(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns RV_TRUE if the lr param exists in the address. Otherwise returns RV_FALSE.

RvSipAddrUrlSetLrParam()

DESCRIPTION

Sets the lr parameter in the *address* URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetLrParam(
    IN RvSipAddressHandle    hSipAddr,
    IN RvBool                bLrParam);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

bLrParam

RV_TRUE specifies that the lr exists in the address.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetLrParamType()

DESCRIPTION

Returns the type of the lr param that exists in the *address* (lr; lr-true; lr=1;).

SYNTAX

```
RvSipUrlLrType RvSipAddrUrlGetLrParamType(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the RvSipUrlLrType.

RvSipAddrUrlSetLrParamType()

DESCRIPTION

Sets the lr parameter type in the *address* URL object (lr; lr-true; lr=1;).

SYNTAX

```
RvStatus RvSipAddrUrlSetLrParamType(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipUrlLrType        eLrParamType);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

eLrParamType

The type of lr parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetMethod()

DESCRIPTION

Gets the method type enumeration from the URL *address*. If this function returns RVSIP_METHOD_OTHER, call RvSipAddrUrlGetStrMethod() to get the actual method in a string format.

SYNTAX

```
RvSipMethodType RvSipAddrUrlGetMethod(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the method type enumeration from the URL *address*.

RvSipAddrUrlGetStrMethod()

DESCRIPTION

Retrieves the method type string value from the URL *address*. If the value of *bufferLen* is big enough, the function will copy the parameter into the *strBuffer*. Otherwise, it will return RV_ERROR_INSUFFICIENT_BUFFER, and the *actualLen* parameter will contain the needed buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrMethod(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hHeader

The handle of the URL *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter +1, for a NULL value at the end.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetMethod()

DESCRIPTION

Sets the method type in the URL *address*. If *eMethodType* is RVSIP_METHOD_OTHER, *strMethodType* is copied to the header. Otherwise *strMethodType* is ignored.

SYNTAX

```
RvStatus RvSipAddrUrlSetMethod(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipMethodType       eMethodType,
    IN RvChar*               pMethodTypeStr);
```

PARAMETERS

hHeader

The handle of the URL *address*.

eMethodType

The method type to be set in the object.

strMethodType

You can use this parameter only if the *eMethodType* parameter is set to RVSIP_METHOD_OTHER. In this case, you can supply the method as a string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetTransport()

DESCRIPTION

Gets the transport protocol enumeration value from the URL object. If the returned Transport enumeration is RVSIP_TRANSPORT_OTHER, you can get the transport protocol string by calling RvSipAddrUrlGetStrTransport().

SYNTAX

```
RvSipTransport RvSipAddrUrlGetTransport(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the transportType enumeration value.

REMARKS

If this function returns RVSIP_METHOD_OTHER, call RvSipAddrUrlGetStrTransport().

RvSipAddrUrlGetStrTransport()

DESCRIPTION

Copies the transport protocol string from the URL *address* into a given buffer. Use this function if RvSipAddrUrlGetTransport() returns RVSIP_TRANSPORT_OTHER. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrTransport(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetTransport()

DESCRIPTION

Sets the transport protocol value in the URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetTransport(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipTransport        eTransport,
    IN RvChar*               strTransport);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

eTransport

The transport protocol enumeration value to be set in the object.

strTransport

You can use this parameter only if the eTransport parameter is set to RVSIP_TRANSPORT_OTHER. In this case you can supply the transport protocol as a string.

RETURN VALUES

Returns RvStatus.

REMARKS

If eTransport is RVSIP_TRANSPORT_OTHER, strTransport is copied to the URL addressobject, otherwise strTransport is ignored.

RvSipAddrUrlGetMaddrParam()

DESCRIPTION

Copies the maddr parameter from the *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the actualLen parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetMaddrParam(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetMaddrParam()

DESCRIPTION

Sets the value of the maddr parameter in the URL object or removes the maddr parameter from the object.

SYNTAX

```
RvStatus RvSipAddrUrlSetMaddrParam(
    IN RvSipAddressHandle    hSipUrl,
    IN RvChar*               strMaddrParam);
```

PARAMETERS

hSipUrl

The handle of the URL *address*.

strMaddrParam

The maddr parameter to be set in the object. If NULL is supplied, the maddr parameter is removed from the URL object.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetUserParam()

DESCRIPTION

Gets the user parameter enumeration value from the URL object. If *eUserParam* is RVSIP_USERPARAM_OTHER, then you can get the user parameter string value using RRvSipAddrUrlGetStrUserParam().

SYNTAX

```
RvSipUserParam RvSipAddrUrlGetUserParam(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

The enumeration value of the requested userParam.

REMARKS

If this function returns RVSIP_USERPARAM_OTHER, call RvSipAddrUrlGetStrUserParam() to get the user parameter as a string.

RvSipAddrUrlGetStrUserParam()

DESCRIPTION

Copies the user param string from the URL *address* into a given buffer. Use this function if RvSipAddrUrlGetUserParam() returns RVSIP_USERPARAM_OTHER. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrUserParam(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetUserParam()

DESCRIPTION

Sets the user param value within the URL object. If *eUserParam* is set to RVSIP_USERPARAM_OTHER, the string in *strUserParam* is set to the URL user parameter. Otherwise, *strUserParam* is ignored.

SYNTAX

```
RvStatus RvSipAddrUrlSetUserParam(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipUserParam        eUserParam,
    IN RvChar*               strUserParam);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

eUserParam

The enumeration value of the userParam.

strUserParam

String containing the user param when eUserParam value is RVSIP_USERPARAM_OTHER.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetHeaders()

DESCRIPTION

Copies the headers parameter of the *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetHeaders(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

Buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetHeaders()

DESCRIPTION

Used to set the headers parameter in the URL *address*.

SYNTAX

```
RvStatus RvSipAddrUrlSetHeaders(
    IN RvSipAddressHandle    hSipUrl,
    IN RvChar*               strHeaders);
```

PARAMETERS

hSipUrl

The handle of the URL *address*.

strHeaders

The headers parameter to be set in the object. If NULL is supplied, the existing headers parameter is removed.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetHeadersList()

DESCRIPTION

Takes the URL headers parameter as a string, and parses it into a list of headers. The application must supply a list structure and a consecutive buffer with the URL headers parameter string.

SYNTAX

```
RvStatus RvSipAddrUrlGetHeadersList(
    IN RvSipAddressHandle       hSipAddr,
    IN RvSipCommonListHandle    hHeadersList,
    IN RvChar*                  pHeadersBuffer);
```

PARAMETERS

hSipAddr

The handle of the *address*.

hHeadersList

The handle of the headers list object.

pHeadersBuffer

The buffer containing a textual URL headers parameter string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetHeadersList()

DESCRIPTION

Sets the headers parameter from the headers list. This function encodes all headers. During encoding, each header coverts reserved characters to its escaped form. Each header also sets an equal sign (=) instead of a colon (:) after the header name. This function also sets an ampersand (&) between headers. If you wish the headers in the header list to take compact form, you should mark each of the headers to be compact form.

SYNTAX

```
RvStatus RvSipAddrUrlSetHeadersList(
    IN RvSipAddressHandle       hSipAddr,
    IN RvSipCommonListHandle    hHeadersList);
```

PARAMETERS

hSipAddr

The handle of the *address*.

hHeadersList

The handle of the header list object.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetCompParam()

DESCRIPTION

Gets the compression type enumeration value from the URL object. If the returned compression enumeration is RVSIP_COMP_OTHER, you can get the compression type string by calling RvSipAddrUrlGetStrCompParam().

SYNTAX

```
RvSipCompType RvSipAddrUrlGetCompParam(
    IN RvSipAddressHandle hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the compression type enumeration value.

RvSipAddrUrlGetStrCompParam()

DESCRIPTION

Copies the compression type string from the URL *address* into a given buffer. Use this function if RvSipAddrUrlGetCompParam() returns RVSIP_COMP_OTHER. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, this function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrCompParam(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

The buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetCompParam()

DESCRIPTION

Sets the compression type value in the URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetCompParam(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipCompType         eComp,
    IN RvChar*               strCompParam);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

eComp

The compression type enumeration value to be set in the object.

strCompParam

You can use this parameter only if the *eComp* parameter is set to RVSIP_COMP_OTHER. In this case, you can supply the compression type as a string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetOtherParams()

DESCRIPTION

Copies the URL other-params parameter of the *address* into a given buffer. Not all the *address* parameters have separated parameters in the *address*. Parameters with no specific parameters are referred to as “other-params. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetOtherParams(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

The buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the *address*. Not all the *address* parameters have separated parameters in the *address*. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipAddrUrlSetOtherParams(
    IN RvSipAddressHandle    hSipUrl,
    IN RvChar*               strUrlParams);
```

PARAMETERS

hSipUrl

The handle of the URL *address*.

strUrlParams

The extended parameters to be set in the *address*. If NULL is supplied, the existing extended parameters are removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlIsSecure()

DESCRIPTION

Specifies whether or not the URL *address* scheme is secure. (The URL *address* is secure with the “sips:” prefix and is not secure with the “sip:” prefix.)

SYNTAX

```
RvBool RvSipAddrUrlIsSecure(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns whether or not the URL is secure.

RvSipAddrUrlSetIsSecure()

DESCRIPTION

Sets the URL *address* to be secure or not secure. (The URL *address* is secure if it set with the “sips:” prefix and not secure if set with the “sip:” prefix.)

SYNTAX

```
RvStatus RvSipAddrUrlSetIsSecure(
    IN RvSipAddressHandle    hSipAddr,
    IN RvBool                bIsSecure);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

bIsSecure

Indicates whether or not the URL is secure.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetCPC()

DESCRIPTION

Gets the CPC enumeration value from the URL object. If the returned CPC enumeration is RVSIP_CPC_TYPE_OTHER, you can get the CPC protocol string by calling RvSipAddrUrlGetStrCPC().

SYNTAX

```
RvSipUriCPCType RvSipAddrUrlGetCPC(
    IN RvSipAddressHandle hSipAddr);
```

PARAMETERS

hSipAddr

The handle to the URL *address*.

RETURN VALUES

Returns the CPCType enumeration value.

RvSipAddrUrlGetStrCPC()

DESCRIPTION

Copies the CPC protocol string from the URL *address* into a given buffer. Use this function if RvSipAddrUrlGetCPC() returns RVSIP_CPC_TYPE_OTHER. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrCPC(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle to the URL address object.

strBuffer

The buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetCPC()

DESCRIPTION

Sets the CPC value in the URL *address*.

SYNTAX

```
RvStatus RvSipAddrUrlSetCPC(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipUriCPCType       eCPC,
    IN RvChar*               strCPC);
```

PARAMETERS

hSipAddr

The handle to the URL *address*.

eCPC

The CPC enumeration value to be set in the URL *address*.

strCPC

You can use this parameter only if *eCPC* is set to RVSIP_CPC_TYPE_OTHER. In this case, you can supply the CPC protocol as a string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetStrGr()

DESCRIPTION

Copies the Gr parameter from the URL *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetStrGr(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle to the URL *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetGr()

DESCRIPTION

Gets the Gr parameter from the URL *address*.

SYNTAX

```
RvBool RvSipAddrUrlGetGr(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipUrl

The handle to the URL *address*.

RETURN VALUES

Returns the Gr parameter, or RV_FALSE if missing.

RvSipAddrUrlSetGr()

DESCRIPTION

Sets the value of the Gr parameter in the URL *address*, or removes the Gr parameter from the URL *address*.

SYNTAX

```
RvStatus RvSipAddrUrlSetGr(
    IN RvSipAddressHandle    hSipAddr,
    IN RvBool                bGrActive,
    IN RvChar*               strGr);
```

PARAMETERS

hSipUrl

The handle to the URL *address*.

bGrActive

A Boolean stating whether or not the gr parameter exists (can be empty).

strGr

The Gr parameter to be set in the URL *address*. If NULL is supplied, the Gr parameter is removed from the URL *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlSetSigCompIdParam()

DESCRIPTION

Sets the sigcomp-id value in the URL object.

SYNTAX

```
RvStatus RvSipAddrUrlSetSigCompIdParam(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strSigCompIdParam);
```

PARAMETERS

hSipAddr

The handle to the URL *address* object.

strSigCompIdParam

The sigcomp-id string.

RETURN VALUES

Returns RvStatus.

RvSipAddrUrlGetSigCompIdParam()

DESCRIPTION

Copies the sigcomp-id parameter from the *address* object into a given buffer. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrUrlGetSigCompIdParam(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle to the URL *address* object.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

Absolute URI Get and Set Functions

The Absolute URI Get and Set functions are:

- RvSipAddrAbsUriGetScheme()
- RvSipAddrAbsUriSetScheme()
- RvSipAddrAbsUriGetIdentifier()
- RvSipAddrAbsUriSetIdentifier()

RvSipAddrAbsUriGetScheme()

DESCRIPTION

Copies the scheme parameter from the *address* into a given buffer. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, this function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrAbsUriGetScheme(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the absolute URI *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrAbsUriSetScheme()

DESCRIPTION

Sets the scheme string in the absolute URI *address*.

SYNTAX

```
RvStatus RvSipAddrAbsUriSetScheme(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strScheme);
```

PARAMETERS

hSipAddr

The handle of the absolute URI *address*.

strScheme

The scheme to be set in the object. If NULL is supplied, the existing scheme is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrAbsUriGetIdentifier()

DESCRIPTION

Copies the identifier parameter from the absolute URI *address* into a given buffer. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, this function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrAbsUriGetIdentifier(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the absolute URI *address*.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrAbsUriSetIdentifier()

DESCRIPTION

Sets the identifier string in the absolute URI *address*.

SYNTAX

```
RvStatus RvSipAddrAbsUriSetIdentifier(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strIdentifier);
```

PARAMETERS

hSipAddr

The handle of the absolute URI *address*.

strIdentifier

The identifier string to be set in the object. If NULL is supplied, the existing identifier is removed from the *address*.

RETURN VALUES

Returns RvStatus.

TEL URI GET AND SET FUNCTIONS

The TEL URI Get and Set functions are:

- RvSipAddrTelUriSetIsGlobalPhoneNumber()
- RvSipAddrTelUriGetIsGlobalPhoneNumber()
- RvSipAddrTelUriSetPhoneNumber()
- RvSipAddrTelUriGetPhoneNumber()
- RvSipAddrTelUriSetExtension()
- RvSipAddrTelUriGetExtension()
- RvSipAddrTelUriSetIsdnSubAddr()
- RvSipAddrTelUriGetIsdnSubAddr()
- RvSipAddrTelUriSetPostDial()
- RvSipAddrTelUriGetPostDial()
- RvSipAddrTelUriSetContext()
- RvSipAddrTelUriGetContext()
- RvSipAddrTelUriSetEnumdiParamType()
- RvSipAddrTelUriGetEnumdiParamType()
- RvSipAddrTelUriSetOtherParams()
- RvSipAddrTelUriGetOtherParams()

RvSipAddrTelUriSetIsGlobalPhoneNumber()

DESCRIPTION

Indicates whether or not this TEL URL *address* is a global phone number.

SYNTAX

```
RvStatus RvSipAddrTelUriSetIsGlobalPhoneNumber(
    IN RvSipAddressHandle    hSipAddr,
    IN RvBool                bIsGlobalPhoneNumber);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

bIsGlobalPhoneNumber

RV_TRUE indicates that this TEL URI represents a global phone number. Otherwise, RV_FALSE.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetIsGlobalPhoneNumber()

DESCRIPTION

Returns RV_TRUE if this *address* indicates a global phone number.

SYNTAX

```
RvBool RvSipAddrTelUriGetIsGlobalPhoneNumber(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

RETURN VALUES

Returns RV_TRUE if this *address* indicates a global phone number.

RvSipAddrTelUriSetPhoneNumber()

DESCRIPTION

Sets the phone-number string in the TEL URI *address*.

SYNTAX

```
RvStatus RvSipAddrTelUriSetPhoneNumber(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strPhoneNumber);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strPhoneNumber

The phone-number string to be set in the object. If NULL is supplied, the existing phone-number is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetPhoneNumber()

DESCRIPTION

Copies the phone-number parameter from the TEL URI *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetPhoneNumber(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
   OUT RvUint*                actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriSetExtension()

DESCRIPTION

Sets the extension string in the TEL URI *address*.

SYNTAX

```
RvStatus RvSipAddrTelUriSetExtension(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strExtension);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strExtension

The extension string to be set in the object. If NULL is supplied, the existing extension is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetExtension()

DESCRIPTION

Copies the extension parameter from the TEL URI *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetExtension(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriSetIsdnSubAddr()

DESCRIPTION

Sets the isdn-sub-address string in the TEL URI *address*.

SYNTAX

```
RvStatus RvSipAddrTelUriSetIsdnSubAddr(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strIsdnSubAddr);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strIsdnSubAddr

The Isdn-sub-address string to be set in the object. If NULL is supplied, the existing Isdn-sub-address is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetIsdnSubAddr()

DESCRIPTION

Copies the isdn-sub-address parameter from the TEL URI *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetIsdnSubAddr(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strBuffer,
    IN RvUint                bufferLen,
    OUT RvUint*              actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriSetPostDial()

DESCRIPTION

Sets the post-dial string in the TEL URI *address*.

SYNTAX

```
RvStatus RvSipAddrTelUriSetPostDial(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strPostDial);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strPostDial

The post-dial string to be set in the object. If NULL is supplied, the existing post-dial is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetPostDial()

DESCRIPTION

Copies the post-dial parameter from the TEL URI *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetPostDial(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriSetContext()

DESCRIPTION

Sets the context string in the TEL URI *address*.

SYNTAX

```
RvStatus RvSipAddrTelUriSetContext(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strContext);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strContext

The context string to be set in the object. If NULL is supplied, the existing context is removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetContext()

DESCRIPTION

Copies the context parameter from the TEL URI *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetContext(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriSetEnumdiParamType()

DESCRIPTION

Sets the enumdi parameter type in a TEL URI *address* (undefined or exists and empty).

SYNTAX

```
RvStatus RvSipAddrTelUriSetEnumdiParamType(
    IN RvSipAddressHandle       hSipAddr,
    IN RvSipTelUriEnumdiType    eEnumdiType);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

eEnumdiType

The type of the enumdi parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetEnumdiParamType()

DESCRIPTION

Returns the type of the enumdi param in a TEL URI address (undefined or exists and empty).

SYNTAX

```
RvSipTelUriEnumdiType RvSipAddrTelUriGetEnumdiParamType(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

RETURN VALUES

Returns RvSipTelUriEnumdiType.

RvSipAddrTelUriSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the TEL URI *address*. Not all the *address* parameters have separated parameters in the *address*. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipAddrTelUriSetOtherParams(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strOtherParams);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strOtherParams

The other-params string to be set in the object. If NULL is supplied, the existing other-params are removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrTelUriGetOtherParams()

DESCRIPTION

Copies the URL other-params parameter of the *address* into a given buffer. Not all the *address* parameters have separated parameters in the *address*. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrTelUriGetOtherParams(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the TEL URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

DIAMETER GET AND SET FUNCTIONS

The Diameter Get and Set functions are:

- RvSipAddrDiameterUriSetIsSecure()
- RvSipAddrDiameterUriSetProtocolType()
- RvSipAddrDiameterUriGetProtocolType()
- RvSipAddrDiameterUriGetHost()
- RvSipAddrDiameterUriSetHost()
- RvSipAddrDiameterUriGetPortNum()
- RvSipAddrDiameterUriSetPortNum()
- RvSipAddrDiameterUriGetTransport()
- RvSipAddrDiameterUriGetStrTransport()
- RvSipAddrDiameterUriSetTransport()
- RvSipAddrDiameterUriSetOtherParams()
- RvSipAddrDiameterUriGetOtherParams()

RvSipAddrDiameterUriSetIsSecure()

DESCRIPTION

Sets the secure flag in the URL *address*.

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetIsSecure(
    IN RvSipAddressHandle    hSipAddr,
    IN RvBool                bIsSecure);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

bIsSecure

Indicates whether or not the URL *address* is secure. (For example, sips/not secured).

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriSetProtocolType()

DESCRIPTION

Sets the protocol type in a DIAMETER URI *address* object (undefined or exists and is empty).

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetProtocolType(
    IN RvSipAddressHandle       hSipAddr,
    IN RvSipDiameterProtocol    eProtocolType);
```

PARAMETERS

hSipAddr

The handle of the DIAMETER URI *address*.

eProtocolType

The type of the protocol parameter.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriGetProtocolType()

DESCRIPTION

Returns the type of the protocol in a DIAMETER URI *address* (undefined or exists and is empty).

SYNTAX

```
RvSipDiameterProtocol RvSipAddrDiameterUriGetProtocolType(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the DIAMETER URI *address*.

RETURN VALUES

Returns RvSipDiameterProtocol.

RvSipAddrDiameterUriGetHost()

DESCRIPTION

Copies the host field from the *address* into a given buffer. If the *bufferLen* is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrDiameterUriGetHost(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriSetHost()

DESCRIPTION

Sets the host in the URL *address*, or removes the existing host from the URL *address*.

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetHost(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strHost);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strHost

The host value to be set in the *address*. If NULL is supplied, the existing host is removed.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriGetPortNum()

DESCRIPTION

Gets the port number from the DIAMETER URI *address*.

SYNTAX

```
RvInt32 RvSipAddrDiameterUriGetPortNum(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

SipUrl

The handle of the DIAMETER URI *address*.

RETURN VALUES

Returns the port number, or UNDEFINED if the port is not set.

RvSipAddrDiameterUriSetPortNum()

DESCRIPTION

Sets the port number in the URL *address*.

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetPortNum(
    IN RvSipAddressHandle    hSipAddr,
    IN RvInt32               portNum);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

portNum

The port number value to be set in the *address*. To remove the port number from the *address*, you can set this parameter to UNDEFINED.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriGetTransport()

DESCRIPTION

Gets the transport protocol enumeration value from the URL *address*. If the returned transport enumeration is RVSIP_TRANSPORT_OTHER, you can get the transport protocol string by calling RvSipAddrDiameterUriGetStrTransport(). If this function returns RVSIP_METHOD_OTHER, call RvSipAddrDiameterUriGetStrTransport().

SYNTAX

```
RvSipTransport RvSipAddrDiameterUriGetTransport(
    IN RvSipAddressHandle    hSipAddr);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

RETURN VALUES

Returns the transport type enumeration value.

RvSipAddrDiameterUriGetStrTransport()

DESCRIPTION

Copies the transport protocol string from the URL *address* into a given buffer. Use this function if RvSipAddrDiameterUriGetTransport() returns RVSIP_TRANSPORT_OTHER. If the *bufferLen* is adequate, this function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrDiameterUriGetStrTransport(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

strBuffer

The buffer to include the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriSetTransport()

DESCRIPTION

Sets the transport protocol value in the URL *address*.

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetTransport(
    IN RvSipAddressHandle    hSipAddr,
    IN RvSipTransport        eTransport,
    IN RvChar*               strTransport);
```

PARAMETERS

hSipAddr

The handle of the URL *address*.

eTransport

The transport protocol enumeration value to be set in the *address*.

strTransport

You can use this parameter only if the *eTransport* parameter is set to RVSIP_TRANSPORT_OTHER. In this case you can supply the transport protocol as a string.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriSetOtherParams()

DESCRIPTION

Sets the “other params” field in the DIAMETER URI *address*. Not all the *address* parameters have separated fields in the *address* object. Parameters with no specific fields are referred to as “other params”. They are kept in the *address* in one concatenated string in the form: name=value;name=value...

SYNTAX

```
RvStatus RvSipAddrDiameterUriSetOtherParams(
    IN RvSipAddressHandle    hSipAddr,
    IN RvChar*               strOtherParams);
```

PARAMETERS

hSipAddr

The handle of the DIAMETER URI *address*.

strOtherParams

The “other params” string to be set in the *address*. If NULL is supplied, the existing “other params” are removed from the *address*.

RETURN VALUES

Returns RvStatus.

RvSipAddrDiameterUriGetOtherParams()

DESCRIPTION

Copies the URL “other params” field of the *address* into a given buffer. Not all the *address* parameters have separated fields in the *address* object. Parameters with no specific fields are referred to as “other params”. They are kept in the *address* in one concatenated string in the form: name=value;name=value... If the *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, this function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAddrDiameterUriGetOtherParams(
    IN  RvSipAddressHandle    hSipAddr,
    IN  RvChar*               strBuffer,
    IN  RvUint                bufferLen,
    OUT RvUint*               actualLen);
```

PARAMETERS

hSipAddr

The handle of the DIAMETER URI *address*.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the given buffer.

actualLen

The length of the requested parameter + 1, to include a NULL character at the end of the string.

RETURN VALUES

Returns RvStatus.

Diameter Get and Set Functions
RvSipAddrDiameterUriGetOtherParams()

3

ALLOW HEADER FUNCTIONS

The Allow header functions enable you to construct, copy, encode, parse, access and change Allow header parameters. This section contains Allow header functions found in the *RvSipAllowHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAllowHeaderConstructInMsg()
- RvSipAllowHeaderConstruct()
- RvSipAllowHeaderCopy()
- RvSipAllowHeaderEncode()
- RvSipAllowHeaderParse()
- RvSipAllowHeaderParseValue()
- RvSipAllowHeaderFix()

RvSipAllowHeaderConstructInMsg()

DESCRIPTION

Constructs an Allow header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the header list.

SYNTAX

```
RvStatus RvSipAllowHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    IN  RvBool                     pushHeaderAtHead,
    OUT RvSipAllowHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Allow header.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Allow header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAllowHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipAllowHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Allow header.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderCopy()

DESCRIPTION

Copies all parameters from a source Allow header to a destination Allow header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAllowHeaderCopy(
    INOUT RvSipAllowHeaderHandle    hDestination,
    IN    RvSipAllowHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Allow header.

hSource

The handle of the source Allow header.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderEncode()

DESCRIPTION

Encodes an Allow header to a textual Allow header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAllowHeaderEncode(
    IN  RvSipAllowHeaderHandle    hHeader,
    IN  HRPOOL                    hPool,
    OUT HPAGE*                    hPage,
    OUT RvUint32*                 length);
```

PARAMETERS

hHeader

The handle of the Allow header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipAllowHeaderParse()

DESCRIPTION

Parses a SIP textual Allow header into an Allow header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipAllowHeaderParse(
    IN RvSipAllowHeaderHandle    hHeader,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

A handle of the Allow header.

buffer

The buffer containing a textual Allow header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct an Allow header before using this function.

RvSipAllowHeaderParseValue()

DESCRIPTION

Parses a SIP textual Allow header value into an Allow header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipAllowHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipAllowHeaderParseValue(
    IN RvSipAllowHeaderHandle    hHeader,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

The handle of the Allow header.

buffer

The buffer containing a textual Allow header value.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderFix()

DESCRIPTION

Fixes an Allow header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipAllowHeaderFix(
    IN RvSipAllowHeaderHandle    hHeader,
    IN RvChar*                   pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAllowHeaderGetStringLength()
- RvSipAllowHeaderGetMethodType()
- RvSipAllowHeaderGetStrMethodType()
- RvSipAllowHeaderSetMethodType()
- RvSipAllowHeaderGetStrBadSyntax()
- RvSipAllowHeaderSetStrBadSyntax()

RvSipAllowHeaderGetStringLength()

DESCRIPTION

Some of the Allow header parameters are kept in a string format, for example, the method string. In order to get such a parameter from the Allow header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAllowHeaderGetStringLength(
    IN RvSipAllowHeaderHandle        hHeader,
    IN RvSipAllowHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Allow header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipAllowHeaderGetMethodType()

DESCRIPTION

Gets the method type enumeration from the Allow header.

SYNTAX

```
RvSipMethodType RvSipAllowHeaderGetMethodType(
    IN RvSipAllowHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Allow header.

RETURN VALUES

Returns the method type enumeration from the header.

REMARKS

If this function returns RVSIP_METHOD_OTHER, call RvSipAllowHeaderGetStrMethodType(), to get the actual method in a string format.

RvSipAllowHeaderGetStrMethodType()

DESCRIPTION

Copies the method type string from the Allow header into a given buffer. Use this function if RvSipAllowHeaderGetMethodType() returns RVSIP_METHOD_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAllowHeaderGetStrMethodType(
    IN  RvSipAllowHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the Allow header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderSetMethodType()

DESCRIPTION

Sets the method type in the Allow header.

SYNTAX

```
RvStatus RvSipAllowHeaderSetMethodType(
    IN RvSipAllowHeaderHandle    hHeader,
    IN RvSipMethodTypeeM         ethodType,
    IN RvChar*                   strMethodType);
```

PARAMETERS

hHeader

The handle of the Allow header.

eMethodType

The method type to be set in the object.

strMethodType

You can use this parameter only if the eMethodType parameter is set to RVSIP_METHOD_OTHER. In this case, you can supply the method as a string.

RETURN VALUES

Returns RvStatus.

REMARKS

If eMethodType is RVSIP_METHOD_OTHER, strMethodType is copied to the header. Otherwise strMethodType is ignored.

RvSipAllowHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Allow header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipAllowHeaderGetStrBadSyntax(
    IN  RvSipAllowHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAllowHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. Using this function, you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Allow header.

SYNTAX

```
RvStatus RvSipAllowHeaderSetStrBadSyntax(
    IN RvSipAllowHeaderHandle    hHeader,
    IN RvChar*                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipAllowHeaderSetStrBadSyntax()

4

ALLOW-EVENTS HEADER FUNCTIONS

The Allow-Events header functions enable you to construct, copy, encode, parse, access and change Allow-Events header parameters. This section contains Allow-Events header functions found in the *RvSipAllowEventsHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAllowEventsHeaderConstructInMsg()
- RvSipAllowEventsHeaderConstruct()
- RvSipAllowEventsHeaderCopy()
- RvSipAllowEventsHeaderEncode()
- RvSipAllowEventsHeaderParse()
- RvSipAllowEventsHeaderParseValue()
- RvSipAllowEventsHeaderFix()

RvSipAllowEventsHeaderConstructInMsg()

DESCRIPTION

Constructs an Allow-Events header inside a given *message*. The header is kept in the header list of the message.You can choose to insert the header either at the head or tail of the header list.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderConstructInMsg(
    IN  RvSipMsgHandle                  hSipMsg,
    IN  RvBool                          pushHeaderAtHead,
    OUT RvSipAllowEventsHeaderHandle*   hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE), or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Allow-Events header.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Allow-Events header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderConstruct(
    IN  RvSipMsgMgrHandle                hMsgMgr,
    IN  HRPOOL                           hPool,
    IN  HPAGE                            hPage,
    OUT RvSipAllowEventsHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Allow-Events header.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderCopy()

DESCRIPTION

Copies all parameters from a source Allow-Events header to a destination Allow-Events header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderCopy(
    INOUT RvSipAllowEventsHeaderHandle    hDestination,
    IN    RvSipAllowEventsHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Allow-Events header.

hSource

The handle of the source Allow-Events header.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderEncode()

DESCRIPTION

Encodes an Allow-Events header to a textual Allow-Events header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAllowEventsHeaderEncode(
    IN  RvSipAllowEventsHeaderHandle    hHeader,
    IN  HRPOOL                          hPool,
    OUT HPAGE*                          hPage,
    OUT RvUint32*                       length);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderParse()

DESCRIPTION

Parses a SIP textual Allow-Events header into an Allow-Events header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderParse(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         buffer);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

buffer

The buffer containing a textual Allow-Events header.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderParseValue()

DESCRIPTION

Parses a SIP textual Allow-Events header value into an Allow-Events header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipAllowEventsHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderParseValue(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         buffer);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

buffer

The buffer containing a textual Allow-Events header value.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderFix()

DESCRIPTION

Fixes a Allow-Events header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderFix(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAllowEventsHeaderGetStringLength()
- RvSipAllowEventsHeaderGetRpoolString()
- RvSipAllowEventsHeaderSetRpoolString()
- RvSipAllowEventsHeaderGetEventPackage()
- RvSipAllowEventsHeaderSetEventPackage()
- RvSipAllowEventsHeaderGetEventTemplate()
- RvSipAllowEventsHeaderSetEventTemplate()
- RvSipAllowEventsHeaderGetStrBadSyntax()
- RvSipAllowEventsHeaderSetStrBadSyntax()
- RvSipAllowEventsHeaderSetCompactForm()
- RvSipAllowEventsHeaderGetCompactForm()

RvSipAllowEventsHeaderGetStringLength()

DESCRIPTION

Some of the Allow-Events header parameters are kept in a string format. In order to get such a parameter from the Allow-Events header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAllowEventsHeaderGetStringLength(
    IN RvSipAllowEventsHeaderHandle        hHeader,
    IN RvSipAllowEventsHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipAllowEventsHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Allow-Events header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderGetRpoolString(
 IN RvSipAllowEventsHeaderHandle      hSipAllowEventsHeader,
 IN RvSipAllowEventsHeaderStringName  eStringName,
 INOUT RPOOL_Ptr                      *pRpoolPtr);
```

PARAMETERS

hSipAllowEventsHeader

The handle of the Allow-Events header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Allow-Events header.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Allow-Events header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderSetRpoolString(
  IN RvSipAllowEventsHeaderHandle     hSipAllowEventsHeader,
  IN RvSipAllowEventsHeaderStringName eStringName,
  IN RPOOL_Ptr                        *pRpoolPtr);
```

PARAMETERS

hSipAllowEventsHeader

The handle of the Allow-Events header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderGetEventPackage()

DESCRIPTION

Copies the event-package string from the Allow-Events header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderGetEventPackage(
    IN  RvSipAllowEventsHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderSetEventPackage()

DESCRIPTION

Sets the event-package in the Allow-Events header.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderSetEventPackage(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         strEventPackage);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

strEventPackage

The event string.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderGetEventTemplate()

DESCRIPTION

Copies the Event template string from the Allow-Events header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderGetEventTemplate(
    IN  RvSipAllowEventsHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderSetEventTemplate()

DESCRIPTION

Sets the Event template in the Allow-Events header.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderSetEventTemplate(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         strEvent);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

strEvent

The event string.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad AllowEvents header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderGetStrBadSyntax(
    IN  RvSipAllowEventsHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal AllowEvents header.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderSetStrBadSyntax(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvChar*                         strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderSetCompactForm(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvBool                          bIsCompact);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

bIsCompact

Specifies whether or not to use a compact form.

RETURN VALUES

Returns RvStatus.

RvSipAllowEventsHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the header.

SYNTAX

```
RvStatus RvSipAllowEventsHeaderGetCompactForm(
    IN RvSipAllowEventsHeaderHandle    hHeader,
    IN RvBool                          *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

pbIsCompact

Specifies whether or not to use a compact form.

RETURN VALUES

Returns RvStatus.

5

AUTHENTICATION HEADER FUNCTIONS

Authentication header functions enable you to construct, copy, encode, parse, access and change Authentication header parameters. This chapter contains Authentication header functions found in the *RvSipAuthenticationHeader.h* header file.

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAuthenticationHeaderConstructInMsg()
- RvSipAuthenticationHeaderConstruct()
- RvSipAuthenticationHeaderCopy()
- RvSipAuthenticationHeaderEncode()
- RvSipAuthenticationHeaderParse()
- RvSipAuthenticationHeaderParseValue()
- RvSipAuthenticationHeaderFix()

RvSipAuthenticationHeaderConstructInMsg()

DESCRIPTION

Constructs an Authentication header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderConstructInMsg(
    IN  RvSipMsgHandle                    hSipMsg,
    IN  RvBool                            pushHeaderAtHead,
    OUT RvSipAuthenticationHeaderHandle   *hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Authentication header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Authentication header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderConstruct(
    IN  RvSipMsgMgrHandle                   hMsgMgr,
    IN  HRPOOL                              hPool,
    IN  HPAGE                               hPage,
    OUT RvSipAuthenticationHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Authentication header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderCopy()

DESCRIPTION

Copies all parameters from a source Authentication header to a destination Authentication header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderCopy(
    IN RvSipAuthenticationHeaderHandle    hDestination,
    IN RvSipAuthenticationHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Authentication header.

hSource

The handle of the source Authentication header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderEncode()

DESCRIPTION

Encodes an Authentication header to a textual Authentication header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAuthenticationHeaderEncode(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  HRPOOL                             hPool,
    OUT HPAGE*                             hPage,
    OUT RvUint32*                          length);
```

PARAMETERS

hHeader

The handle of the Authentication header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipAuthenticationHeaderParse()

DESCRIPTION

Parses a SIP textual Authentication header into an Authentication header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderParse(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            buffer);
```

PARAMETERS

hHeader

A handle of the Authentication header.

buffer

The buffer containing a textual Authentication header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct an Authentication header before using this function.

RvSipAuthenticationHeaderParseValue()

DESCRIPTION

Parses a SIP textual Authentication header value into an Authentication header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipAuthenticationHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderParseValue(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            buffer);
```

PARAMETERS

hHeader

The handle of the Authentication header.

buffer

The buffer containing a textual Authentication header value.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderFix()

DESCRIPTION

Fixes an Authentication header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderFix(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAuthenticationHeaderGetStringLength()
- RvSipAuthenticationHeaderGetAuthScheme()
- RvSipAuthenticationHeaderGetStrAuthScheme()
- RvSipAuthenticationHeaderSetAuthScheme()
- RvSipAuthenticationHeaderGetRealm()
- RvSipAuthenticationHeaderSetRealm()
- RvSipAuthenticationHeaderGetDomain()
- RvSipAuthenticationHeaderSetDomain()
- RvSipAuthenticationHeaderGetNonce()
- RvSipAuthenticationHeaderSetNonce()
- RvSipAuthenticationHeaderGetOpaque()
- RvSipAuthenticationHeaderSetOpaque()
- RvSipAuthenticationHeaderGetStrQop()
- RvSipAuthenticationHeaderSetStrQop()
- RvSipAuthenticationHeaderGetStale()
- RvSipAuthenticationHeaderSetStale()
- RvSipAuthenticationHeaderGetAuthAlgorithm()
- RvSipAuthenticationHeaderGetStrAuthAlgorithm()
- RvSipAuthenticationHeaderSetAuthAlgorithm()
- RvSipAuthenticationHeaderGetQopOption()
- RvSipAuthenticationHeaderSetQopOption()
- RvSipAuthenticationHeaderGetHeaderType()
- RvSipAuthenticationHeaderSetHeaderType()
- RvSipAuthenticationHeaderGetOtherParams()
- RvSipAuthenticationHeaderSetOtherParams()
- RvSipAuthenticationHeaderGetRpoolString()
- RvSipAuthenticationHeaderSetRpoolString()
- RvSipAuthenticationHeaderGetStrBadSyntax()
- RvSipAuthenticationHeaderSetStrBadSyntax()
- RvSipAuthenticationHeaderGetAKAVersion()
- RvSipAuthenticationHeaderSetAKAVersion()
- RvSipAuthenticationHeaderGetStrIntegrityKey()

- RvSipAuthenticationHeaderSetStrIntegrityKey()
- RvSipAuthenticationHeaderGetStrCipherKey()
- RvSipAuthenticationHeaderSetStrCipherKey()

RvSipAuthenticationHeaderGetStringLength()

DESCRIPTION

Some of the Authentication header parameters are kept in a string format, for example, the nonce parameter. In order to get such a parameter from the Authentication header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAuthenticationHeaderGetStringLength(
    IN RvSipAuthenticationHeaderHandle        hHeader,
    IN RvSipAuthenticationHeaderStringName    eStringName);
```

PARAMETERS

hHeader

The handle of the Authentication header.

eStringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of specified string.

RvSipAuthenticationHeaderGetAuthScheme()

DESCRIPTION

Gets the authentication scheme enumeration from the Authentication header.

SYNTAX

```
RvSipAuthScheme RvSipAuthenticationHeaderGetAuthScheme(
    IN RvSipAuthenticationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authentication header.

RETURN VALUES

Returns the authentication scheme enumeration from the header.

REMARKS

If this function returns RVSIP_AUTH_SCHEME_OTHER, call RvSipAuthenticationHeaderGetStrAuthScheme() to get the actual Authentication scheme in a string format.

RvSipAuthenticationHeaderGetStrAuthScheme()

DESCRIPTION

Copies the authentication scheme string value from the authentication object into a given buffer. Use this function if RvSipAuthenticationHeaderGetAuthScheme() returns RVSIP_AUTH_SCHEME_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrAuthScheme(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT  RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetAuthScheme()

DESCRIPTION

Sets the authentication scheme in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetAuthScheme(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvSipAuthScheme                    eAuthScheme,
    IN RvChar*                            strAuthScheme);
```

PARAMETERS

hHeader

The handle of the Authentication header.

eAuthScheme

The authentication scheme to be set in the object.

strAuthScheme

The authentication scheme to be set in the object. You can only use this parameter if eAuthScheme is RVSIP_AUTH_SCHEME_OTHER. Otherwise, the parameter is ignored.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetRealm()

DESCRIPTION

Copies the realm value of the Authentication header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetRealm(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetRealm()

DESCRIPTION

Sets the realm string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetRealm(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pRealm);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pRealm

The realm to be set in the Authentication header. If a NULL value is supplied, the existing realm string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetDomain()

DESCRIPTION

Copies the domain value of the authentication object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetDomain(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetDomain()

DESCRIPTION

Sets the domain string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetDomain(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pDomain);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pDomain

The domain to be set in the Authentication header. If a NULL value is supplied, the existing domain string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetNonce()

DESCRIPTION

Copies the nonce value of the authentication object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetNonce(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetNonce()

DESCRIPTION

Sets the nonce string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetNonce(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pNonce);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pNonce

The nonce to be set in the Authentication header. If a NULL value is supplied, the existing nonce string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetOpaque()

DESCRIPTION

Copies the opaque value of the authentication object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetOpaque(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetOpaque()

DESCRIPTION

Sets the opaque string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetOpaque(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pOpaque);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pOpaque

The opaque string to be set in the Authentication header. If a NULL value is supplied, the existing opaque string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetStrQop()

DESCRIPTION

Copies the Qop string value of the authentication object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrQop(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetStrQop()

DESCRIPTION

Sets the Qop string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetStrQop(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pQop);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pQop

The Qop string to be set in the Authentication header. If a NULL value is supplied, the existing Qop string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetStale()

DESCRIPTION

Gets the stale enumeration from the Authentication header.

SYNTAX

```
RvSipAuthStale RvSipAuthenticationHeaderGetStale(
    IN RvSipAuthenticationHeaderHandle    hSipAuthHeader);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

RETURN VALUES

Returns the stale enumeration value from the Authentication header.

RvSipAuthenticationHeaderSetStale()

DESCRIPTION

Sets the stale parameter in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetStale(
    IN RvSipAuthenticationHeaderHandle    hSipAuthHeader,
    IN RvSipAuthStale                     eStale);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

eStale

The stale value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetAuthAlgorithm()

DESCRIPTION

Gets the algorithm type enumeration from the Authentication header.

SYNTAX

```
RvSipAuthAlgorithm
  RvSipAuthenticationHeaderGetAuthAlgorithm(
    IN RvSipAuthenticationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authentication header.

RETURN VALUES

Returns the algorithm type enumeration of the header.

REMARKS

If this function returns RVSIP_AUTH_ALGORITHM_OTHER, call RvSipAuthenticationHeaderGetStrAuthAlgorithm(), to get the actual algorithm type in a string format.

RvSipAuthenticationHeaderGetStrAuthAlgorithm()

DESCRIPTION

Copies the authentication algorithm string from the Authentication header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrAuthAlgorithm(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetAuthAlgorithm()

DESCRIPTION

Sets the authentication algorithm in authentication object.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetAuthAlgorithm(
    IN RvSipAuthenticationHeaderHandle   Header,
    IN RvSipAuthAlgorithm                eAuthAlgorithm,
    IN RvChar                            *strAuthAlgorithm);
```

PARAMETERS

hHeader

The handle of the Authentication header.

eAuthAlgorithem

The authentication algorithm to be set in the object.

strAuthAlgorithm

The authentication algorithm string to be set in the object. You can use this parameter only if eAuthAlgorithm is set to RVSIP_AUTH_ALGORITHM_OTHER. Otherwise, the parameter is ignored.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetQopOption()

DESCRIPTION

Gets the Qop option from the Authentication header.

SYNTAX

```
RvSipAuthQopOption RvSipAuthenticationHeaderGetQopOption(
    IN RvSipAuthenticationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authentication header.

RETURN VALUES

The Qop option from the header.

RvSipAuthenticationHeaderSetQopOption()

DESCRIPTION

Sets the Qop option enumeration in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetQopOption(
    IN RvSipAuthenticationHeaderHandle    hSipAuthHeader,
    IN RvSipAuthQopOption eQop);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

eQop

The Qop option enumeration to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetHeaderType()

DESCRIPTION

Gets the header type enumeration from the Authentication header.

SYNTAX

```
RvSipAuthenticationHeaderType
  RvSipAuthenticationHeaderGetHeaderType(
    IN RvSipAuthenticationHeaderHandle    hSipAuthHeader);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

RETURN VALUES

Returns the Authentication header type enumeration from the header.

RvSipAuthenticationHeaderSetHeaderType()

DESCRIPTION

Sets the header type in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetHeaderType(
    IN RvSipAuthenticationHeaderHandle    hSipAuthHeader,
    IN RvSipAuthenticationHeaderType      eHeaderType);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

eHeaderType

The header type to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetOtherParams()

DESCRIPTION

Copies the Authentication header other-params parameter of the Authentication header into a given buffer. Not all the Authentication header parameters have separated parameters in the Authentication header. Parameters with no specific parameters are referred to as other-parameters. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetOtherParams(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Authentication header. Not all the Authentication header parameters have separated parameters in the Authentication header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetOtherParams(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pOtherParams);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pOtherParams

The other-params parameter to be set in the Authentication header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Authentication header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetRpoolString(
    IN    RvSipAuthenticationHeaderHandle
                        hSipAuthHeader,
    IN    RvSipAuthenticationHeaderStringName
                        eStringName,
    INOUT RPOOL_Ptr
                        *pRpoolPtr);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Authentication header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Authentication header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetRpoolString(
    IN RvSipAuthenticationHeaderHandle       hSipAuthHeader,
    IN RvSipAuthenticationHeaderStringName   eStringName,
    IN RPOOL_Ptr                             *pRpoolPtr);
```

PARAMETERS

hSipAuthHeader

The handle of the Authentication header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string.

If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Authentication header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrBadSyntax(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetStrBadSyntax(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RvSipAuthenticationHeaderGetAKAVersion()

DESCRIPTION

Gets the AKA version from the Authentication header.

SYNTAX

```
RvInt32 RvSipAuthenticationHeaderGetAKAVersion(
    IN RvSipAuthenticationHeaderHandle
                                    hSipAuthenticationHeader);
```

PARAMETERS

hAuthenticationHeader

The handle of the Authentication header.

RETURN VALUES

Returns the AKAVersion, or UNDEFINED if the AKAVersion does not exist.

RvSipAuthenticationHeaderSetAKAVersion()

DESCRIPTION

Sets the AKA version number of the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetAKAVersion(
    IN RvSipAuthenticationHeaderHandle
                                    hSipAuthenticationHeader,
    IN RvInt32                      AKAVersion);
```

PARAMETERS

hAuthenticationHeader

The handle of the Authentication header.

AKAVersion

The AKA version number to be set in the header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetStrIntegrityKey()

DESCRIPTION

Copies the IntegrityKey string value of the Authentication header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrIntegrityKey(
    IN  RvSipAuthenticationHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetStrIntegrityKey()

DESCRIPTION

Sets the IntegrityKey string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetStrIntegrityKey(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pIntegrityKey);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pIntegrityKey

The IntegrityKey string to be set in the Authentication header. If a NULL value is supplied, the existing IntegrityKey string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderGetStrCipherKey()

DESCRIPTION

Copies the CipherKey string value of the Authentication object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderGetStrCipherKey(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            strBuffer,
    IN RvUint                             bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationHeaderSetStrCipherKey()

DESCRIPTION

Sets the CipherKey string in the Authentication header.

SYNTAX

```
RvStatus RvSipAuthenticationHeaderSetStrCipherKey(
    IN RvSipAuthenticationHeaderHandle    hHeader,
    IN RvChar*                            pCipherKey);
```

PARAMETERS

hHeader

The handle of the Authentication header.

pCipherKey

The CipherKey string to be set in the Authentication header. If a NULL value is supplied, the existing CipherKey string in the header is removed.

RETURN VALUES

Returns RvStatus.

6

AUTHENTICATION-INFO HEADER FUNCTIONS

Authentication-Info header functions enable you to construct, copy, encode, parse, access and change Authentication-Info header parameters. This chapter contains Authentication-Info header functions found in the *RvSipAuthenticationInfoHeader.h* header file.

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAuthenticationInfoHeaderConstructInMsg()
- RvSipAuthenticationInfoHeaderConstruct()
- RvSipAuthenticationInfoHeaderCopy()
- RvSipAuthenticationInfoHeaderEncode()
- RvSipAuthenticationInfoHeaderParse()
- RvSipAuthenticationInfoHeaderParseValue()
- RvSipAuthenticationInfoHeaderFix()

RvSipAuthenticationInfoHeaderConstructInMsg()

DESCRIPTION

Constructs an Authentication-Info header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderConstructInMsg(
  IN  RvSipMsgHandle                      hSipMsg,
  IN  RvBool                              pushHeaderAtHead,
  OUT RvSipAuthenticationInfoHeaderHandle *hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE), or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Authentication-Info header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Authentication-Info header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderConstruct(
    IN  RvSipMsgMgrHandle                    hMsgMgr,
    IN  HRPOOL                               hPool,
    IN  HPAGE                                hPage,
    OUT RvSipAuthenticationInfoHeaderHandle  *hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Authentication-Info header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderCopy()

DESCRIPTION

Copies the informative field from a source Authentication-Info header to a destination Authentication-Info header. The informative field is the field indicated by the RvSipAuthInfoType enumeration.You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderCopy(
    IN RvSipAuthenticationInfoHeaderHandle    hDestination,
    IN RvSipAuthenticationInfoHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Authentication-Info header.

hSource

The handle of the source Authentication-Info header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderEncode()

DESCRIPTION

Encodes an Authentication-Info header to a textual Authentication-Info header. The textual header is placed on a page taken from a specified pool. In order to copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderEncode(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  HRPOOL                                 hPool,
    OUT HPAGE*                                 hPage,
    OUT RvUint32*                              length);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderParse()

DESCRIPTION

Parses a SIP textual Authentication-Info header into an Authentication-Info header. All the textual fields are placed inside the object.

Note Before performing the parse operation the stack resets all the header fields. Therefore, if the parse function fails, you will not be able to get the previous header field values.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderParse(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar*                                buffer);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

buffer

Buffer containing a textual Authentication-Info header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderParseValue()

DESCRIPTION

Parses a SIP textual Authentication-Info header value into an Authentication-Info header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object.All the textual fields are placed inside the object.

Note Use the RvSipAuthenticationInfoHeaderParse() function to parse strings that also include the header-name.

Note Before performing the parse operation the stack resets all the header fields. Therefore, if the parse function fails, you will not be able to get the previous header field values.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderParseValue(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar*                                buffer);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

buffer

The buffer containing a textual Authentication-Info header value.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderFix()

DESCRIPTION

Fixes an Authentication-Info header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header, which parses a given correct header-value string to the supplied header. If parsing succeeds, this function places all fields inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderFix(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar*                                pFixedBuffer);
```

PARAMETERS

Input: hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAuthenticationInfoHeaderGetStringLength()
- RvSipAuthenticationInfoHeaderGetNextNonce()
- RvSipAuthenticationInfoHeaderSetNextNonce()
- RvSipAuthenticationInfoHeaderGetCNonce()
- RvSipAuthenticationInfoHeaderSetCNonce()
- RvSipAuthenticationInfoHeaderGetResponseAuth()
- RvSipAuthenticationInfoHeaderSetResponseAuth()
- RvSipAuthenticationInfoHeaderGetNonceCount()
- RvSipAuthenticationInfoHeaderSetNonceCount()
- RvSipAuthenticationInfoHeaderGetQopOption()
- RvSipAuthenticationInfoHeaderGetStrQop()
- RvSipAuthenticationInfoHeaderSetQopOption()
- RvSipAuthenticationInfoHeaderGetStrBadSyntax()
- RvSipAuthenticationInfoHeaderSetStrBadSyntax()
- RvSipAuthenticationInfoHeaderGetRpoolString()
- RvSipAuthenticationInfoHeaderSetRpoolString()

RvSipAuthenticationInfoHeaderGetStringLength()

DESCRIPTION

Some of the Authentication-Info header fields are kept in a string format, for example, the Authentication-Info header next-nonce string. To get such a field from the Authentication-Info header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAuthenticationInfoHeaderGetStringLength(
  IN RvSipAuthenticationInfoHeaderHandle       hHeader,
  IN RvSipAuthenticationInfoHeaderStringName   eStringName);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

eStringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipAuthenticationInfoHeaderGetNextNonce()

DESCRIPTION

Copies the next-nonce string from the Authentication-Info header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetNextNonce(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetNextNonce()

DESCRIPTION

Sets the next-nonce string in the Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetNextNonce(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar                                 *pNextNonce);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

pNextNonce

The next-nonce string to be set in the Authentication-Info header. If NULL is supplied, the existing next-nonce is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetCNonce()

DESCRIPTION

Copies the cnonce string from the Authentication-Info header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetCNonce(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetCNonce()

DESCRIPTION

Sets the cnonce string in the Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetCNonce(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar*                                pCNonce);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

pCNonce

The cnonce string to be set in the Authentication-Info header. If a NULL value is supplied, the existing cnonce string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetResponseAuth()

DESCRIPTION

Copies the response-auth string from the Authentication-Info header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetResponseAuth(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetResponseAuth()

DESCRIPTION

Sets the response-auth string in the Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetResponseAuth(
   IN RvSipAuthenticationInfoHeaderHandle    hHeader,
   IN RvChar*                                pResponseAuth);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

pResponseAuth

The response-auth string to be set in the Authentication-Info header. If a NULL value is supplied, the existing response-auth string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetNonceCount()

DESCRIPTION

Gets the nonce count value from the Authentication-Info header.

SYNTAX

```
RvInt32 RvSipAuthenticationInfoHeaderGetNonceCount(
 IN RvSipAuthenticationInfoHeaderHandle hSipAuthInfoHeader);
```

PARAMETERS

hSipAuthInfoHeader

The handle of the Authentication-Info header.

RETURN VALUES

Returns the nonce count value.

RvSipAuthenticationInfoHeaderSetNonceCount()

DESCRIPTION

Sets the nonce count value in the Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetNonceCount(
  IN RvSipAuthenticationInfoHeaderHandle hSipAuthInfoHeader,
  IN RvInt32                             nonceCount);
```

PARAMETERS

hSipAuthInfoHeader

The handle of the Authentication-Info header.

nonceCount

The nonce count value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetQopOption()

DESCRIPTION

Gets the Qop option enumeration from the Authentication-Info header.

SYNTAX

```
RvSipAuthQopOption
RvSipAuthenticationInfoHeaderGetQopOption(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader);
```

PARAMETERS

hHeader

Returns the Qop option enumeration from the object.

RETURN VALUES

Returns the Qop type from the object.

RvSipAuthenticationInfoHeaderGetStrQop()

DESCRIPTION

Copies the Qop string value of the Authentication-Info object into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetStrQop(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetQopOption()

DESCRIPTION

Sets the Qop string in the Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetQopOption(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvSipAuthQopOption                     eQop,
    IN RvChar*                                strQop);
```

PARAMETERS

hHeader

The handle of the Authentication-Info header.

eQop

The Qop option enumeration value to be set in the object.

strQop

You can use this parameter only if the *eQop* parameter is set to RVSIP_AUTH_QOP_OTHER. In this case you can supply the Qop option as a string.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Authentication-Info header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetStrBadSyntax(
    IN  RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with bad-syntax. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Authentication-Info header.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetStrBadSyntax(
    IN RvSipAuthenticationInfoHeaderHandle    hHeader,
    IN RvChar*                                strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Authentication-Info header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderGetRpoolString(
 IN    RvSipAuthenticationInfoHeaderHandle
                                          hSipAuthInfoHeader,
 IN    RvSipAuthenticationInfoHeaderStringName
                                          eStringName,
 INOUT RPOOL_Ptr
                                          *pRpoolPtr);
```

PARAMETERS

hSipAuthInfoHeader

The handle of the Authentication-Info header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an rpool. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Authentication-Info header.

RETURN VALUES

Returns RvStatus.

RvSipAuthenticationInfoHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Authentication-Info header. The given string is located on an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderSetRpoolString(
  IN RvSipAuthenticationInfoHeaderHandle
                                          hSipAuthInfoHeader,
  IN RvSipAuthenticationInfoHeaderStringName
                                          eStringName,
  IN RPOOL_Ptr
                                          *pRpoolPtr);
```

PARAMETERS

hSipAuthInfoHeader

The handle of the Authentication-Info header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an rpool where the new string is located.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipAuthenticationInfoHeaderSetRpoolString()

7

AUTHORIZATION HEADER FUNCTIONS

The Authorization header functions enable you to construct, copy, encode, parse, access and change Authorization header parameters. This section contains Authorization header functions found in the *RvSipAuthorizationHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipAuthorizationHeaderConstructInMsg()
- RvSipAuthorizationHeaderConstruct()
- RvSipAuthorizationHeaderCopy()
- RvSipAuthorizationHeaderEncode()
- RvSipAuthorizationHeaderParse()
- RvSipAuthorizationHeaderParseValue()
- RvSipAuthorizationHeaderFix()

RvSipAuthorizationHeaderConstructInMsg()

DESCRIPTION

Constructs an Authorization header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderConstructInMsg(
    IN  RvSipMsgHandle                    hSipMsg,
    IN  RvBool                            pushHeaderAtHead,
    OUT RvSipAuthorizationHeaderHandle   *hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Authorization header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Authorization header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderConstruct(
    IN  RvSipMsgMgrHandle                 hMsgMgr,
    IN  HRPOOL                            hPool,
    IN  HPAGE                             hPage,
    OUT RvSipAuthorizationHeaderHandle   *hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Authorization header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderCopy()

DESCRIPTION

Copies all parameters from a source Authorization header to a destination Authorization header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderCopy(
    IN RvSipAuthorizationHeaderHandle       hDestination,
    IN RvSipAuthorizationHeaderHandle       hSource);
```

PARAMETERS

hDestination

The handle of the destination Authorization header.

hSource

The handle of the source Authorization header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderEncode()

DESCRIPTION

Encodes an Authorization header to a textual Authorization header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipAuthorizationHeaderEncode(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  HRPOOL                            hPool,
    OUT HPAGE*                            hPage,
    OUT RvUint32*                         length);
```

PARAMETERS

hHeader

The handle of the Authorization header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipAuthorizationHeaderParse()

DESCRIPTION

Parses a SIP textual Authorization header into an Authorization header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderParse(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           buffer);
```

PARAMETERS

hHeader

The handle of the Authorization header.

buffer

The buffer containing a textual Authorization header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct an Authorization header before using this function.

RvSipAuthorizationHeaderParseValue()

DESCRIPTION

Parses a SIP textual Authorization header value into an Authorization header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipAuthorizationHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderParseValue(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           buffer);
```

PARAMETERS

hHeader

The handle of the Authorization header.

buffer

The buffer containing a textual Authorization header value.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderFix()

DESCRIPTION

Fixes an Authorization header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderFix(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipAuthorizationHeaderGetStringLength()
- RvSipAuthorizationHeaderGetCredentialIdentifier()
- RvSipAuthorizationHeaderGetAuthScheme()
- RvSipAuthorizationHeaderGetStrAuthScheme()
- RvSipAuthorizationHeaderSetAuthScheme()
- RvSipAuthorizationHeaderGetRealm()
- RvSipAuthorizationHeaderSetRealm()
- RvSipAuthorizationHeaderGetNonce()
- RvSipAuthorizationHeaderSetNonce()
- RvSipAuthorizationHeaderGetOpaque()
- RvSipAuthorizationHeaderSetOpaque()
- RvSipAuthorizationHeaderGetUserName()
- RvSipAuthorizationHeaderSetUserName()
- RvSipAuthorizationHeaderGetNonceCount()
- RvSipAuthorizationHeaderSetNonceCount()
- RvSipAuthorizationHeaderGetAuthAlgorithm()
- RvSipAuthorizationHeaderGetStrAuthAlgorithm()
- RvSipAuthorizationHeaderSetAuthAlgorithm()
- RvSipAuthorizationHeaderGetQopOption()
- RvSipAuthorizationHeaderSetQopOption()
- RvSipAuthorizationHeaderGetResponse()
- RvSipAuthorizationHeaderSetResponse()
- RvSipAuthorizationHeaderGetCNonce()
- RvSipAuthorizationHeaderSetCNonce()
- RvSipAuthorizationHeaderGetDigestUri()
- RvSipAuthorizationHeaderSetDigestUri()
- RvSipAuthorizationHeaderGetHeaderType()
- RvSipAuthorizationHeaderSetHeaderType()
- RvSipAuthorizationHeaderGetOtherParams()
- RvSipAuthorizationHeaderSetOtherParams()
- RvSipAuthorizationHeaderSetRpoolString()
- RvSipAuthorizationHeaderGetRpoolString()

- RvSipAuthorizationHeaderGetStrBadSyntax()
- RvSipAuthorizationHeaderSetStrBadSyntax()
- RvSipAuthorizationHeaderGetStrAuts()
- RvSipAuthorizationHeaderSetStrAuts()
- RvSipAuthorizationHeaderGetIntegrityProtected()
- RvSipAuthorizationHeaderSetIntegrityProtected()

RvSipAuthorizationHeaderGetStringLength()

DESCRIPTION

Some of the Authorization header parameters are kept in a string format, for example, the Authorization header user name string. In order to get such a parameter from the Authorization header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipAuthorizationHeaderGetStringLength(
    IN RvSipAuthorizationHeaderHandle        hHeader,
    IN RvSipAuthorizationHeaderStringName    eStringName);
```

PARAMETERS

hHeader

The handle of the Authorization header.

eStringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipAuthorizationHeaderGetCredentialIdentifier()

DESCRIPTION

Gets the identifiers of the creditable Authentication header. This function will return the realm and username strings without quotation marks (""). If you want these parameters in quotation marks, use the specific Get functions, and not this function. If the *realmLen* or *usernameLen* is adequate, the function copies the strings into *strRealm* and *strUsername* buffers, Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *realmActualLen*, and *usernameActualLen* contain the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetCredentialIdentifier(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strRealmBuff,
    IN  RvUint                            realmBuffLen,
    IN  RvChar*                           strUsernameBuff,
    IN  RvUint                            usernameBuffLen,
    IN  RvChar*                           strNonceBuff,
    IN  RvUint                            nonceBuffLen,
    OUT RvUint*                           realmActualLen,
    OUT RvUint*                           usernameActualLen,
    OUT RvUint*                           nonceActualLen);
```

PARAMETERS

hHeader

The handle of the header.

strRealmBuff

The buffer with which to fill the requested parameter.

realmBuffLen

The length of the given realm buffer.

strUsernameBuff

The buffer to fill with the username parameter.

usernameBuffLen

The length of the username given buffer.

strNonceBuff

The buffer to fill with the nonce parameter.

nonceBuffLen

The length of the nonce given buffer.

realmActualLen

The length of the realm parameter +1 to include a NULL value at the end.

usernameActualLen

The length of the username parameter +1 to include a NULL value at the end.

nonceActualLen

The length of the nonce parameter +1 to include a NULL value at the end.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetAuthScheme()

DESCRIPTION

Gets the authentication scheme enumeration value from the Authorization header.

SYNTAX

```
RvSipAuthScheme RvSipAuthorizationHeaderGetAuthScheme(
    IN RvSipAuthorizationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authorization header.

RETURN VALUES

Returns the authentication scheme enumeration from the object.

RvSipAuthorizationHeaderGetStrAuthScheme()

DESCRIPTION

Gets the authentication scheme string value from the Authorization header. Use this function if RvSipAuthorizationHeaderGetAuthScheme() returns RVSIP_AUTH_SCHEME_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetStrAuthScheme(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetAuthScheme()

DESCRIPTION

Sets the authentication scheme in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetAuthScheme(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvSipAuthScheme                   eAuthScheme,
    IN RvChar*                           strAuthScheme);
```

PARAMETERS

hHeader

The handle of the Authorization header.

eAuthScheme

The Authentication scheme to be set in the object.

strAuthScheme

Text string giving the Authentication scheme to be set in the object. Use this parameter only if eAuthScheme is RVSIP_AUTH_SCHEME_OTHER. Otherwise, the parameter is set to NULL.

RETURN VALUES

Returns RvStatus.

REMARKS

If *eAuthScheme* is RVSIP_AUTH_SCHEME_OTHER, *strAuthScheme* is copied to the header. Otherwise, *strAuthScheme* is ignored.

RvSipAuthorizationHeaderGetRealm()

DESCRIPTION

Copies the realm string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetRealm(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetRealm()

DESCRIPTION

Sets the realm string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetRealm(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pRealm);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pRealm

The realm string to be set in the Authorization header. If NULL is supplied, the existing realm is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetNonce()

DESCRIPTION

Copies the nonce string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetNonce(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetNonce()

DESCRIPTION

Sets the nonce string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetNonce(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pNonce);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pNonce

The nonce string to be set in the Authorization header. If a NULL value is supplied, the existing nonce string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetOpaque()

DESCRIPTION

Copies the opaque string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetOpaque(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetOpaque()

DESCRIPTION

Sets the opaque string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetOpaque(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pOpaque);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pOpaque

The opaque string to be set in the Authorization header. If a NULL value is supplied, the existing opaque string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetUserName()

DESCRIPTION

Copies the user name string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetUserName(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetUserName()

DESCRIPTION

Sets the user name string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetUserName(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pUserName);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pUserName

The user name string to be set in the Authorization header. If a NULL value is supplied, the existing user name string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetNonceCount()

DESCRIPTION

Gets the nonce count value from the Authorization header.

SYNTAX

```
RvInt32 RvSipAuthorizationHeaderGetNonceCount(
    IN RvSipAuthorizationHeaderHandle    hSipAuthHeader);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

RETURN VALUES

Returns the nonce count value.

RvSipAuthorizationHeaderSetNonceCount()

DESCRIPTION

Sets the nonce count value in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetNonceCount(
    IN RvSipAuthorizationHeaderHandle    hSipAuthHeader,
    IN RvInt32                           nonceCount);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

nonceCount

The nonce count value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetAuthAlgorithm()

DESCRIPTION

Gets the authentication algorithm enumeration from the Authorization header.

SYNTAX

```
RvSipAuthAlgorithm RvSipAuthorizationHeaderGetAuthAlgorithm(
    IN RvSipAuthorizationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authorization header.

RETURN VALUES

Returns the Authentication algorithm enumeration from the header.

REMARKS

If this function returns RVSIP_AUTH_ALGORITHM_OTHER, call RvSipAuthorizationHeaderGetStrAuthAlgorithm() to get the algorithm in a string format.

RvSipAuthorizationHeaderGetStrAuthAlgorithm()

DESCRIPTION

Copies the authentication algorithm string from the Authorization header into a given buffer. Use this function if RvSipAuthorizationHeaderGetAuthAlgorithm() returns RVSIP_AUTH_ALGORITHM_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetStrAuthAlgorithm(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetAuthAlgorithm()

DESCRIPTION

Sets the authentication algorithm in Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetAuthAlgorithm(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvSipAuthAlgorithm                eAuthAlgorithm,
    IN RvChar                            *strAuthAlgorithm);
```

PARAMETERS

hHeader

The handle of the Authorization header.

eAuthAlgorithm

The Authentication algorithm to be set in the object.

strAuthAlgorithm

Text string giving the Authorization algorithm to be set in the object. You can use this parameter if eAuthAlgorithm is set to RVSIP_AUTH_ALGORITHM_OTHER. Otherwise, the parameter is set to NULL.

RETURN VALUES

Returns RvStatus.

REMARKS

If *eAuthAlgorithm* is RVSIP_AUTH_SCHEME_OTHER, *strAuthAlgorithm* is copied to the header. Otherwise, *strAuthAlgorithm* is ignored.

RvSipAuthorizationHeaderGetQopOption()

DESCRIPTION

Gets the Qop option enumeration from the Authorization object.

SYNTAX

```
RvSipAuthQopOption RvSipAuthorizationHeaderGetQopOption(
    IN RvSipAuthorizationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authorization header.

RETURN VALUES

Returns the Qop option enumeration from the object.

RvSipAuthorizationHeaderSetQopOption()

DESCRIPTION

Sets the Qop option enumeration in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetQopOption(
    IN RvSipAuthorizationHeaderHandle    hSipAuthHeader,
    IN RvSipAuthQopOption eQop);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

eQop

The Qop option to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetResponse()

DESCRIPTION

Copies the response string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetResponse(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetResponse()

DESCRIPTION

Sets the response string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetResponse(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pResponse);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pResponse

The Response string to be set in the Authorization header. If a NULL value is supplied, the existing Response string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetCNonce()

DESCRIPTION

Copies the cnonce string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetCNonce(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetCNonce()

DESCRIPTION

Sets the cnonce string in the authorization object.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetCNonce(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pCNonce);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pCNonce

The CNonce string to be set in the Authorization header. If a NULL value is supplied, the existing CNonce string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetDigestUri()

DESCRIPTION

Gets the handle of the digest URI *address* from the Authorization header which holds the digest URI parameter.

SYNTAX

```
RvSipAddressHandle RvSipAuthorizationHeaderGetDigestUri(
    IN RvSipAuthorizationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Authorization header

RETURN VALUES

Returns a handle of the *address*, or NULL if the *address* does not exist.

RvSipAuthorizationHeaderSetDigestUri()

DESCRIPTION

Sets the digest URI parameter in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetDigestUri(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvSipAddressHandle                hDigestUri);
```

PARAMETERS

hHeader

The handle of the Authorization header

hdigestUri

The handle of the *address*. If a NULL is supplied, the existing *address* in the Authorization header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetHeaderType()

DESCRIPTION

Gets the header type enumeration from the Authorization header.

SYNTAX

```
RvSipAuthorizationHeaderType
  RvSipAuthorizationHeaderGetHeaderType(
    IN RvSipAuthorizationHeaderHandle    hSipAuthHeader);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

RETURN VALUES

Returns the Authorization header type enumeration from the Authorization header.

RvSipAuthorizationHeaderSetHeaderType()

DESCRIPTION

Sets the header type in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetHeaderType(
    IN RvSipAuthorizationHeaderHandle    hSipAuthHeader,
    IN RvSipAuthorizationHeaderType      eHeaderType);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

eHeaderType

The header type to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetOtherParams()

DESCRIPTION

Copies the Authorization header other-params parameter of the Authorization header into a given buffer. Not all the Authorization header parameters have separated parameters in the Authorization header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetOtherParams(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Authorization header. Not all the Authorization header parameters have separated parameters in the Authorization header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetOtherParams(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           strOtherParams);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strOtherParams

The other-params parameter to be set in the Authorization header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Authorization header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetRpoolString(
    IN RvSipAuthorizationHeaderHandle        hSipAuthHeader,
    IN RvSipAuthorizationHeaderStringName    eStringName,
    IN RPOOL_Ptr                             *pRpoolPtr);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Authorization header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetRpoolString(
  IN    RvSipAuthorizationHeaderHandle       hSipAuthHeader,
  IN    RvSipAuthorizationHeaderStringName   eStringName,
  INOUT RPOOL_Ptr                            *pRpoolPtr);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Authorization header.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Authorization header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetStrBadSyntax(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetStrBadSyntax(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetStrAuts()

DESCRIPTION

Copies the Auts string from the Authorization header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderGetStrAuts(
    IN  RvSipAuthorizationHeaderHandle    hHeader,
    IN  RvChar*                           strBuffer,
    IN  RvUint                            bufferLen,
    OUT RvUint*                           actualLen);
```

PARAMETERS

hHeader

The handle of the Authorization header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderSetStrAuts()

DESCRIPTION

Sets the nonce string in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetStrAuts(
    IN RvSipAuthorizationHeaderHandle    hHeader,
    IN RvChar*                           pAuts);
```

PARAMETERS

hHeader

The handle of the Authorization header.

pAuts

The Auts string to be set in the Authorization header. If a NULL value is supplied, the existing Auts string in the header is removed.

RETURN VALUES

Returns RvStatus.

RvSipAuthorizationHeaderGetIntegrityProtected()

DESCRIPTION

Gets the IntegrityProtected option enumeration from the Authorization header.

SYNTAX

```
RvSipAuthIntegrityProtected
RvSipAuthorizationHeaderGetIntegrityProtected(
    IN RvSipAuthorizationHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The header from which to get the IntegrityProtected enumeration.

RETURN VALUES

Returns the IntegrityProtected type from the object.

RvSipAuthorizationHeaderSetIntegrityProtected()

DESCRIPTION

Sets the IntegrityProtected enumeration in the Authorization header.

SYNTAX

```
RvStatus RvSipAuthorizationHeaderSetIntegrityProtected(
   IN RvSipAuthorizationHeaderHandle  hSipAuthHeader,
   IN RvSipAuthIntegrityProtected     eIntegrityProtected);
```

PARAMETERS

hSipAuthHeader

The handle of the Authorization header.

eIntegrityProtected

The IntegrityProtected value to be set in the header.

RETURN VALUES

Returns RvStatus.

8

CONTACT HEADER FUNCTIONS

The Contact header functions enable you to construct, copy, encode, parse, access and change Contact header parameters. This section contains Contact header functions found in the *RvSipContactHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipContactHeaderConstructInMsg()
- RvSipContactHeaderConstruct()
- RvSipContactHeaderCopy()
- RvSipContactHeaderEncode()
- RvSipContactHeaderParse()
- RvSipContactHeaderParseValue()
- RvSipContactHeaderFix()
- RvSipContactHeaderIsFeatureTagActive()

RvSipContactHeaderConstructInMsg()

DESCRIPTION

Constructs a Contact header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipContactHeaderConstructInMsg(
    IN  RvSipMsgHandle              hSipMsg,
    IN  RvBool                      pushHeaderAtHead,
    OUT RvSipContactHeaderHandle*   hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Contact header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipContactHeaderConstruct(
    IN  RvSipMsgMgrHandle            hMsgMgr,
    IN  HRPOOL                       hPool,
    IN  HPAGE                        hPage,
    OUT RvSipContactHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderCopy()

DESCRIPTION

Copies all parameters from a source Contact header to a destination Contact header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipContactHeaderCopy(
    INOUT RvSipContactHeaderHandle    hDestination,
    IN    RvSipContactHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Contact header.

hSource

The handle of the source Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderEncode()

DESCRIPTION

Encodes a Contact header to a textual Contact header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipContactHeaderEncode(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  HRPOOL                      hPool,
    OUT HPAGE*                      hPage,
    OUT RvUint32*                   length);
```

PARAMETERS

hHeader

The handle of the Contact header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipContactHeaderParse()

DESCRIPTION

Parses a SIP textual Contact header into a Contact header. All the textual parameters are placed inside the object. For example, “Contact:sip:172.20.5.3:5060”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipContactHeaderParse(
    INOUT RvSipContactHeaderHandle    hHeader,
    IN    RvChar*                     buffer);
```

PARAMETERS

hHeader

A handle of the Contact header.

buffer

The buffer containing a textual Contact header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Contact header before using this function.

RvSipContactHeaderParseValue()

DESCRIPTION

Parses a SIP textual Contact header value into an Contact header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipContactHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipContactHeaderParseValue(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar*                     buffer);
```

PARAMETERS

hHeader

The handle of the Contact header.

buffer

The buffer containing a textual Contact header value.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderFix()

DESCRIPTION

Fixes a Contact header with bad-syntax information. A SIP header has the following grammar, “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipContactHeaderFix(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar*                     pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderIsFeatureTagActive()

DESCRIPTION

Checks if the requested feature tag is active.

SYNTAX

```
RvStatus RvSipContactHeaderIsFeatureTagActive(
    IN  RvSipContactHeaderHandle      hHeader,
    IN  RvSipContactFeatureTagType    eType,
    OUT RvBool*                       bIsFeatureActive);
```

PARAMETERS

hHeader

The handle of the header object.

eType

The enumeration denoting the type of feature tag.

bIsFeatureActive

A Boolean denoting whether or not the requested tag is active.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipContactHeaderGetStringLength()
- RvSipContactHeaderGetDisplayName()
- RvSipContactHeaderSetDisplayName()
- RvSipContactHeaderGetAddrSpec()
- RvSipContactHeaderSetAddrSpec()
- RvSipContactHeaderGetExpires()
- RvSipContactHeaderSetExpires()
- RvSipContactHeaderGetQVal()
- RvSipContactHeaderSetQVal()
- RvSipContactHeaderGetAction()
- RvSipContactHeaderSetAction()
- RvSipContactHeaderSetStar()
- RvSipContactHeaderIsStar()
- RvSipContactHeaderGetOtherParams()
- RvSipContactHeaderSetOtherParams()
- RvSipContactHeaderGetRpoolString()
- RvSipContactHeaderSetRpoolString()
- RvSipContactHeaderGetStrBadSyntax()
- RvSipContactHeaderSetStrBadSyntax()
- RvSipContactHeaderSetCompactForm()
- RvSipContactHeaderGetCompactForm()
- RvSipContactHeaderGetStrFeatureTag()
- RvSipContactHeaderSetStrFeatureTag()
- RvSipContactHeaderSetFeatureTagActivation()
- RvSipContactHeaderGetPubGruu()
- RvSipContactHeaderSetPubGruu()
- RvSipContactHeaderGetTempGruu()
- RvSipContactHeaderSetTempGruu()
- RvSipContactHeaderGetRegIDNum()
- RvSipContactHeaderSetRegIDNum()

RvSipContactHeaderGetStringLength()

DESCRIPTION

Some of the Contact header parameters are kept in a string format, for example, the Contact header display name. In order to get such a parameter from the Contact header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipContactHeaderGetStringLength(
    IN RvSipContactHeaderHandle        hHeader,
    IN RvSipContactHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Contact header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipContactHeaderGetDisplayName()

DESCRIPTION

Copies the display name from the Contact header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetDisplayName(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetDisplayName()

DESCRIPTION

Sets the display name in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetDisplayName(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar*                     strDisplayName);
```

PARAMETERS

hHeader

The handle of the header.

strDisplayName

The display name to be set in the Contact header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetAddrSpec()

DESCRIPTION

The address-spec parameter is held in the Contact header as an *address*. This function returns the handle of the *address*.

SYNTAX

```
RvSipAddressHandle RvSipContactHeaderGetAddrSpec(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Contact header.

RETURN VALUES

Returns a handle of the address-spec object, or NULL if the address-spec object does not exist.

RvSipContactHeaderSetAddrSpec()

DESCRIPTION

Sets the “address-spec” parameter in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetAddrSpec(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvSipAddressHandle          hAddrSpec);
```

PARAMETERS

hHeader

The handle of the Contact header.

hAddrSpec

The handle of the address-spec *address*. If NULL is supplied, the existing “address spec” is removed from the Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetExpires()

DESCRIPTION

The Contact header contains an Expires header. This function returns the handle of the expires header.

SYNTAX

```
RvSipExpiresHeaderHandle RvSipContactHeaderGetExpires(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Contact header.

RETURN VALUES

Returns the handle of the expires header, or NULL if the expires header does not exist.

RvSipContactHeaderSetExpires()

DESCRIPTION

Sets the expires header in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetExpires(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvSipExpiresHeaderHandle    hExpires);
```

PARAMETERS

hHeader

The handle of the Contact header.

hExpires

The handle of the expires header to be set in the Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetQVal()

DESCRIPTION

Copies the q parameter from the Contact header into a given buffer. If the value of *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetQVal(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter +1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetQVal()

DESCRIPTION

Sets the q parameter value in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetQVal(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar                      *strQVal);
```

PARAMETERS

hHeader

The handle of the Contact header.

strQVal

The value of the offset of the “q parameter to be set.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetAction()

DESCRIPTION

The Contact header contains the action type. This function returns this type.

SYNTAX

```
RvSipContactAction RvSipContactHeaderGetAction(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Contact header.

RETURN VALUES

Returns the type of the action in the Contact header.

RvSipContactHeaderSetAction()

DESCRIPTION

Sets the action type in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetAction(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvSipContactAction          action);
```

PARAMETERS

hHeader

The handle of the Contact header.

action

The action type to be set.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetStar()

DESCRIPTION

A Contact header can be in the form of “Contact: *”. This function defines the Contact header as an asterisk (*) Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetStar(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderIsStar()

DESCRIPTION

Determines whether or not the Contact header contains an asterisk (*). This means that the header is of the form “Contact: *”.

SYNTAX

```
RvBool RvSipContactHeaderIsStar(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Contact header.

RETURN VALUES

Returns RV_TRUE if the Contact header is of the form “Contact: *”. Otherwise returns RV_FALSE.

RvSipContactHeaderGetOtherParams()

DESCRIPTION

Copies the Contact header other-params parameter of the Contact header into a given buffer. Not all the Contact header parameters have separated parameters in the Contact header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetOtherParams(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the Contact header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Contact header. Not all the Contact header parameters have separated parameters in the Contact header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipContactHeaderSetOtherParams(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar*                     strContactParam);
```

PARAMETERS

hHeader

The handle of the Contact header.

strContactParam

The extended parameters parameter to be set in the Contact header. If NULL is supplied, the existing extended parameters parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Contact header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipContactHeaderGetRpoolString(
    IN    RvSipContactHeaderHandle        hSipContactHeader,
    IN    RvSipContactHeaderStringName    eStringName,
    INOUT RPOOL_Ptr                       *pRpoolPtr);
```

PARAMETERS

hSipContactHeader

The handle of the Contact header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Contact header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Contact header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipContactHeaderSetRpoolString(
    IN RvSipContactHeaderHandle        hSipContactHeader,
    IN RvSipContactHeaderStringName    eStringName,
    IN RPOOL_Ptr                       *pRpoolPtr);
```

PARAMETERS

hSipContactHeader

The handle of the Contact header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Contact header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipContactHeaderGetStrBadSyntax(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetStrBadSyntax(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar*                     strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipContactHeaderSetCompactForm(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvBool                      bIsCompact);
```

PARAMETERS

hHeader

The handle of the Contact header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the header.

SYNTAX

```
RvStatus RvSipContactHeaderGetCompactForm (
    IN RvSipContactHeaderHandle    hHeader,
    IN RvBool                      *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Contact header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetStrFeatureTag()

DESCRIPTION

Copies the feature tag from the Contact header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetStrFeatureTag(
    IN  RvSipContactHeaderHandle      hHeader,
    IN  RvSipContactFeatureTagType    eType,
    IN  RvChar*                       strBuffer,
    IN  RvUint                        bufferLen,
    OUT RvUint*                       actualLen);
```

PARAMETERS

hHeader

The handle of the header object.

eType

The enumeration denoting the type of feature tag.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter, + 1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetStrFeatureTag()

DESCRIPTION

Sets a feature tag in the Contact header object.

SYNTAX

```
RvStatus RvSipContactHeaderSetStrFeatureTag(
    IN RvSipContactHeaderHandle      hHeader,
    IN RvSipContactFeatureTagType    eType,
    IN RvChar*                       pFeatureTag);
```

PARAMETERS

hHeader

The handle of the Contact header object.

eType

The enumeration denoting the type of feature tag.

pFeatureTag

The feature tag to be set in the Contact header. If NULL is supplied, the existing feature tag is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetFeatureTagActivation()

DESCRIPTION

Activates/deactivates a specific feature tag in the Contact header object.

SYNTAX

```
RvStatus RvSipContactHeaderSetFeatureTagActivation(
    IN RvSipContactHeaderHandle      hHeader,
    IN RvSipContactFeatureTagType    eType,
    IN RvBool                        bIsActivated);
```

PARAMETERS

hHeader

The handle of the Contact header object.

eType

The feature tag to be activated in the Contact header.

bIsActivated

A Boolean denoting whether the feature is activated or deactivated.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetPubGruu()

DESCRIPTION

Copies the pub-gruu parameter from the Contact header into a given buffer. If the *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetPubGruu(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle to the Contact header object.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetPubGruu()

DESCRIPTION

Sets the pub-gruu parameter value in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetPubGruu(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar                      *strPubGruu);
```

PARAMETERS

hHeader

The handle to the Contact header object.

strPubGruu

The value of the offset of the pub-gruu parameter to be set.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetTempGruu()

DESCRIPTION

Copies the temp-gruu parameter from the Contact header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContactHeaderGetTempGruu(
    IN  RvSipContactHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT  RvUint*                    actualLen);
```

PARAMETERS

hHeader

The handle to the Contact header object.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderSetTempGruu()

DESCRIPTION

Sets the temp-gruu parameter value in the Contact header.

SYNTAX

```
RvStatus RvSipContactHeaderSetTempGruu(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvChar                      *strTempGruu);
```

PARAMETERS

hHeader

The handle to the Contact header object.

strTempGruu

The value of the offset of the “temp-gruu” parameter to be set.

RETURN VALUES

Returns RvStatus.

RvSipContactHeaderGetRegIDNum()

DESCRIPTION

Gets the reg-id from the Contact header object.

SYNTAX

```
RvInt32 RvSipContactHeaderGetRegIDNum(
    IN RvSipContactHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle to the Contact header object.

RETURN VALUES

Returns the reg-id number, or UNDEFINED if the reg-id does not exist.

RvSipContactHeaderSetRegIDNum()

DESCRIPTION

Sets the reg-id field in the Contact header object.

SYNTAX

```
RvStatus RvSipContactHeaderSetRegIDNum(
    IN RvSipContactHeaderHandle    hHeader,
    IN RvInt32                     regIDNum);
```

PARAMETERS

hHeader

The handle to the Contact header object.

regIDNum

The reg-id value to be set in the object.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipContactHeaderSetRegIDNum()

9

CONTENT-DISPOSITION HEADER FUNCTIONS

Content-Disposition header functions enable you to construct, copy, encode, parse, access and change Content-Disposition header parameters. This section contains Content-Disposition header functions found in the *RvSipContentDipositionHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipContentDispositionHeaderConstructInMsg()
- RvSipContentDispositionHeaderConstructInBodyPart()
- RvSipContentDispositionHeaderConstruct()
- RvSipContentDispositionHeaderCopy()
- RvSipContentDispositionHeaderEncode()
- RvSipContentDispositionHeaderParse()
- RvSipContentDispositionHeaderParseValue()
- RvSipContentDispositionHeaderFix()

RvSipContentDispositionHeaderConstructInMsg()

DESCRIPTION

Constructs a Content-Disposition header inside a given *message*.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderConstructInMsg(
    IN  RvSipMsgHandle                          hSipMsg,
    IN  RvBool                                  pushHeaderAtHead,
    OUT RvSipContentDispositionHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head or tail of the list. If the value is RV_TRUE, the header is pushed to the head of the list. If the value is RV_FALSE, the header is pushed to the tail of the list.

hHeader

The handle of the newly constructed Content-Disposition header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderConstructInBodyPart()

DESCRIPTION

Constructs a Content-Disposition header inside a given message body part object.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderConstructInBodyPart(
    IN  RvSipBodyPartHandle                    hBodyPart,
    OUT RvSipContentDispositionHeaderHandle*   hHeader);
```

PARAMETERS

hBodyPart

The handle of the message body part object.

hHeader

The handle of the newly constructed Content-Disposition header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Content-Disposition header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderConstruct(
    IN  RvSipMsgMgrHandle                       hMsgMgr,
    IN  HRPOOL                                  hPool,
    IN  HPAGE                                   hPage,
    OUT RvSipContentDispositionHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Content-Disposition header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderCopy()

DESCRIPTION

Copies all parameters from a source Content-Disposition header to a destination Content-Disposition header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderCopy(
  INOUT RvSipContentDispositionHeaderHandle    hDestination,
  IN    RvSipContentDispositionHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Content-Disposition header.

hSource

The handle of the source Content-Disposition header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderEncode()

DESCRIPTION

Encodes a Content-Disposition header to a textual Content-Deposition header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipContentDispositionHeaderEncode(
    IN  RvSipContentDispositionHeaderHandle    hHeader,
    IN  HRPOOL                                 hPool,
    OUT HPAGE*                                 hPage,
    OUT RvUint32*                              length);
```

PARAMETERS

hHeader

The handle of the Content-Disposition header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderParse()

DESCRIPTION

Parses a SIP textual Content-Disposition header into a Content-Disposition header. For example, “Content-Disposition: signal; handling=optional”. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderParse(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvChar*                                buffer);
```

PARAMETERS

hHeader

A handle of the Content-Disposition header.

buffer

The buffer containing a textual Content-Deposition header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderParseValue()

DESCRIPTION

Parses a SIP textual Content-Deposition header value into an Content-Disposition header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipContentDispositionHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderParseValue(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvChar*                                buffer);
```

PARAMETERS

hHeader

The handle of the Content-Disposition header.

buffer

The buffer containing a textual Content-Deposition header value.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderFix()

DESCRIPTION

Fixes a Content-Disposition header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderFix(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvChar*                                pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions:

- RvSipContentDispositionHeaderGetStringLength()
- RvSipContentDispositionHeaderGetType()
- RvSipContentDispositionHeaderGetStrType()
- RvSipContentDispositionHeaderSetType()
- RvSipContentDispositionHeaderGetHandling()
- RvSipContentDispositionHeaderGetStrHandling()
- RvSipContentDispositionHeaderSetHandling()
- RvSipContentDispositionHeaderGetOtherParams()
- RvSipContentDispositionHeaderSetOtherParams()
- RvSipContentDispositionHeaderGetStrBadSyntax()
- RvSipContentDispositionHeaderSetStrBadSyntax()

RvSipContentDispositionHeaderGetStringLength()

DESCRIPTION

Some of the Content-Type header parameters are kept in a string format, for example, the handling string. To get such a parameter from the content-type header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipContentDispositionHeaderGetStringLength(
   IN RvSipContentDispositionHeaderHandle       hHeader,
   IN RvSipContentDispositionHeaderStringName   stringName);
```

PARAMETERS

hHeader

The handle of the Content-Disposition header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipContentDispositionHeaderGetType()

DESCRIPTION

Gets the disposition type enumeration value. If RVSIP_DISPOSITIONTYPE_OTHER is returned, you can receive the disposition type string using RvSipContentDispositionHeaderGetStrType().

SYNTAX

```
RvSipDispositionType RvSipContentDispositionHeaderGetType(
    IN RvSipContentDispositionHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the header.

RETURN VALUES

Returns the disposition type enumeration value.

RvSipContentDispositionHeaderGetStrType()

DESCRIPTION

Copies the disposition type string from the Content-Disposition header into a given buffer. Use this function when the disposition type enumeration returned by RvSipContentDispositionHeaderGetType() is RVSIP_DISPOSITIONTYPE_OTHER. If the value for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderGetStrType(
    IN  RvSipContentDispositionHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderSetType()

DESCRIPTION

Sets the disposition type parameter in the Content-Disposition header. If the enumeration given by *eDispType* is RVSIP_DISPOSITIONTYPE_OTHER, this function sets the disposition type string given by *strDispType* in the Content-Type header. Use RVSIP_DISPOSITIONTYPE_OTHER when the disposition type you wish to set to the content-type does not have a matching enumeration value in the RvSipDispositionType enumeration.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderSetType(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvSipDispositionType                   eDispType,
    IN RvChar                                 *strDispType);
```

PARAMETERS

hHeader

The handle of the Content-Disposition header.

eDispType

The disposition type enumeration to be set in the Content-Type header. If RVSIP_DISPOSITIONTYPE_UNDEFINED is supplied, the existing disposition type is removed from the header.

strDispType

The disposition type string to be set in the Content-Type header. This parameter is relevant when *eDispType* is RVSIP_DISPOSITIONTYPE_OTHER.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderGetHandling()

DESCRIPTION

Gets the handling enumeration value. If RVSIP_DISPOSITIONHANDLING_OTHER is returned, you can receive the handling string using RvSipContentDispositionHeaderGetStrHandling().

SYNTAX

```
RvSipDispositionHandling RvSipContentDispositionHeaderGetHandling(
    IN RvSipContentDispositionHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderGetStrHandling()

DESCRIPTION

Copies the handling string from the Content-Disposition header into a given buffer. Use this function when the handling enumeration returned by RvSipContentDispositionHeaderGetHandling() is RVSIP_DISPOSITIONHANDLING_OTHER. If the values for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderGetStrHandling(
    IN  RvSipContentDispositionHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

Length of the buffer.

actualLen

Length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderSetHandling()

DESCRIPTION

Sets the handling parameter in the content-disposition header. If the enumeration given by *eHandling* is RVSIP_DISPOSITIONHANDLING_OTHER, this function sets the handling string given by strHandling in the Content-Type header. Use RVSIP_DISPOSITIONHANDLING_OTHER when the disposition type you wish to set to the content-type does not have a matching enumeration value in the RvSipDispositionHandling enumeration.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderSetHandling(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvSipDispositionHandling               eHandling,
    IN RvChar                                 *strHandling);
```

PARAMETERS

hHeader

The handle of the Content-Disposition header.

eHandling

The handling enumeration to be set in the Content-Type header. If RVSIP_DISPOSITIONHANDLING_UNDEFINED is supplied, the existing handling is removed from the header.

strHandling

The handling string to be set in the Content-Type header. This parameter is relevant when *eHandling* is RVSIP_DISPOSITIONHANDLING_OTHER.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderGetOtherParams()

DESCRIPTION

Not all the Content-Disposition header parameters have separated parameters in the content-disposition header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function copies the Content-Disposition header “other-params” parameter of the content-disposition header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderGetOtherParams(
    IN  RvSipContentDispositionHeaderHandle    hHeader,
    IN  RvChar*                                strBuffer,
    IN  RvUint                                 bufferLen,
    OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderSetOtherParams()

DESCRIPTION

Not all the Content-Disposition header parameters have separated parameters in the content-disposition header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function sets the other-params” parameter in the content-disposition header.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderSetOtherParams(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvChar                                 *otherParams);
```

PARAMETERS

hHeader

The handle of the header.

otherParams

The other-parameters string to be set in the Content-Disposition header. If NULL is supplied, the existing other-parameters string is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Content-Disposition header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderGetStrBadSyntax(
   IN  RvSipContentDispositionHeaderHandle    hHeader,
   IN  RvChar*                                strBuffer,
   IN  RvUint                                 bufferLen,
   OUT RvUint*                                actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentDispositionHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Content-Disposition header.

SYNTAX

```
RvStatus RvSipContentDispositionHeaderSetStrBadSyntax(
    IN RvSipContentDispositionHeaderHandle    hHeader,
    IN RvChar*                                strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

10

CONTENT-TYPE HEADER FUNCTIONS

Content-Type header functions enable you to construct, copy, encode, parse, access and change Content-Type header parameters. This section contains Content-Type header functions found in the *RvSipContentTypeHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipContentTypeHeaderConstructInBody()
- RvSipContentTypeHeaderConstruct()
- RvSipContentTypeHeaderCopy()
- RvSipContentTypeHeaderEncode()
- RvSipContentTypeHeaderParse()
- RvSipContentTypeHeaderParseValue()
- RvSipContentTypeHeaderFix()

RvSipContentTypeHeaderConstructInBody()

DESCRIPTION

Constructs a Content-Type header inside a given body object.

SYNTAX

```
RvStatus RvSipContentTypeHeaderConstructInBody(
    IN  RvSipBodyHandle                 hBody,
    OUT RvSipContentTypeHeaderHandle    *phHeader);
```

PARAMETERS

hBody

The handle of the body object.

phHeader

The handle of the newly constructed Content-Type header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Content-Type header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipContentTypeHeaderConstruct(
    IN  RvSipMsgMgrHandle               hMsgMgr,
    IN  HRPOOL                          hPool,
    IN  HPAGE                           hPage,
    OUT RvSipContentTypeHeaderHandle    *phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

phHeader

The handle of the newly constructed Content-Type header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderCopy()

DESCRIPTION

Copies all parameters from a source Content-Type header to a destination Content-Type header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipContentTypeHeaderCopy(
    INOUT RvSipContentTypeHeaderHandle    hDestination,
    IN    RvSipContentTypeHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Content-Type header.

hSource

The handle of the source Content-Type header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderEncode()

DESCRIPTION

Encodes a Content-Type header to a textual Content-Type header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipContentTypeHeaderEncode(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  HRPOOL                          hPool,
    OUT HPAGE                           *hPage,
    OUT RvUint32                        *length);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderParse()

DESCRIPTION

Parses a SIP textual Content-Type header into a Content-Type header. For example, “Content-Type: multipart/mixed; boundart=unique”. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipContentTypeHeaderParse(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar                          *buffer);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

buffer

The buffer containing a textual Content-Type header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderParseValue()

DESCRIPTION

Parses a SIP textual Content-Type header value into a Content-Type header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipContentTypeHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipContentTypeHeaderParseValue(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar*                         buffer);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

buffer

The buffer containing a textual Content-Type header value.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderFix()

DESCRIPTION

Fixes a Content-Type header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipContentTypeHeaderFix(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar*                         pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipContentTypeHeaderGetStringLength()
- RvSipContentTypeHeaderGetMediaType()
- RvSipContentTypeHeaderGetStrMediaType()
- RvSipContentTypeHeaderSetMediaType()
- RvSipContentTypeHeaderGetMediaSubType()
- RvSipContentTypeHeaderGetStrMediaSubType()
- RvSipContentTypeHeaderSetMediaSubType()
- RvSipContentTypeHeaderGetBoundary()
- RvSipContentTypeHeaderSetBoundary()
- RvSipContentTypeHeaderGetVersion()
- RvSipContentTypeHeaderSetVersion()
- RvSipContentTypeHeaderGetBase()
- RvSipContentTypeHeaderSetBase()
- RvSipContentTypeHeaderGetOtherParams()
- RvSipContentTypeHeaderSetOtherParams()
- RvSipContentTypeHeaderGetStrBadSyntax()
- RvSipContentTypeHeaderSetStrBadSyntax()
- RvSipContentTypeHeaderSetCompactForm()
- RvSipContentTypeHeaderGetCompactForm()
- RvSipContentTypeHeaderGetStart()
- RvSipContentTypeHeaderSetStart()
- RvSipContentTypeHeaderTypeParamGetMediaType()
- RvSipContentTypeHeaderTypeParamGetStrMediaType()
- RvSipContentTypeHeaderTypeParamGetMediaSubType()
- RvSipContentTypeHeaderTypeParamGetStrMediaSubType()
- RvSipContentTypeHeaderTypeParamSetMediaType()
- RvSipContentTypeHeaderTypeParamSetMediaSubType()

RvSipContentTypeHeaderGetStringLength()

DESCRIPTION

Some of the Content-Type header parameters are kept in a string format, such as the Content-Type header parameter boundary. In order to get such a parameter from the Content-Type header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipContentTypeHeaderGetStringLength(
    IN RvSipContentTypeHeaderHandle        hHeader,
    IN RvSipContentTypeHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipContentTypeHeaderGetMediaType()

DESCRIPTION

Gets the media type enumeration value. If RVSIP_MEDIATYPE_OTHER is returned, you can receive the media type string using RvSipContentTypeHeaderGetStrMediaType().

SYNTAX

```
RvSipMediaType RvSipContentTypeHeaderGetMediaType(
    IN RvSipContentTypeHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the header.

RETURN VALUES

Returns the enumeration value of the SIP media type.

RvSipContentTypeHeaderGetStrMediaType()

DESCRIPTION

Copies the media-type string from the Content-Type header into a given buffer. Use this function when the media type enumeration returned by RvSipContentTypeHeaderGetMediaType() is RVSIP_MEDIATYPE_OTHER. If the value for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetStrMediaType(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetMediaType()

DESCRIPTION

Sets the media-type parameter in the Content-Type header. If the enumeration given by *eMediaType* is RVSIP_MEDIATYPE_OTHER, sets the media type string given by *strMediaType* in the Content-Type header. Use RVSIP_MEDIATYPE_OTHER when the media type you wish to set to the Content-Type header does not have a matching enumeration value in the RvSipMediaType enumeration.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetMediaType(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvSipMediaType                  eMediaType,
    IN RvChar                          *strMediaType);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

eMediaType

The media type enumeration to be set in the Content-Type header. If RVSIP_MEDIATYPE_UNDEFINED is supplied, the existing media type is removed from the header.

strMediaType

The media type string to be set in the Content-Type header. (Relevant when *eMediaType* is RVSIP_MEDIATYPE_OTHER).

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetMediaSubType()

DESCRIPTION

Gets the media-sub-type enumeration value. If RVSIP_MEDIASUBTYPE_OTHER is returned, you can receive the media-sub-type string using RvSipContentTypeHeaderGetStrMediaSubType().

SYNTAX

```
RvSipMediaSubType RvSipContentTypeHeaderGetMediaSubType(
    IN RvSipContentTypeHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the header.

RETURN VALUES

Returns the enumeration value of the SIP media-sub-type.

RvSipContentTypeHeaderGetStrMediaSubType()

DESCRIPTION

Copies the media-sub-type string from the Content-Type header into a given buffer. Use this function when the media-sub-type enumeration returned by RvSipContentTypeHeaderGetMediaSubType() is RVSIP_MEDIASUBTYPE_OTHER. If the value for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetStrMediaSubType(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetMediaSubType()

DESCRIPTION

Sets the media-sub-type parameter in the Content-Type header. If the enumeration given by *eMediaSubType* is RVSIP_MEDIASUBTYPE_OTHER, sets the media-sub-type string given by *strMediaSubType* in the Content-Type header. Use RVSIP_MEDIASUBTYPE_OTHER when the media type you wish to set to the Content-Type header does not have a matching enumeration value in the RvSipMediaSubType enumeration.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetMediaSubType(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvSipMediaSubType               eMediaSubType,
    IN RvChar                          *strMediaSubType);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

eMediaSubType

The media-sub-type enumeration to be set in the Content-Type header. If RVSIP_MEDIASUBTYPE_UNDEFINED is supplied, the existing RvSipMediaSubType is removed from the header.

strMediaSubType

The media-sub-type string to be set in the Content-Type header. (Relevant when *eMediaSubType* is RVSIP_MEDIASUBTYPE_OTHER).

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetBoundary()

DESCRIPTION

Copies the boundary string from the Content-Type header into a given buffer. If the value for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetBoundary(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetBoundary()

DESCRIPTION

Sets the boundary string in the Content-Type header. The given string is copied to the Content-Type header.

Note The given boundary may be quoted as defined in IETF RFC 2046.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetBoundary(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar                          *boundary);
```

PARAMETERS

hHeader

The handle of the header.

boundary

The boundary string to be set in the Content-Type header. If NULL is supplied, the existing boundary is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetVersion()

DESCRIPTION

Copies the version string from the Content-Type header into a given buffer. If the value for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetVersion(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetVersion()

DESCRIPTION

Sets the version string in the Content-Type header. The given string is copied to the Content-Type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetVersion(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar                          *version);
```

PARAMETERS

hHeader

The handle of the header.

version

The version string to be set in the Content-Type header. If NULL is supplied, the existing version is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetBase()

DESCRIPTION

Copies the base string from the Content-Type header into a given buffer. If the values for *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetBase(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetBase()

DESCRIPTION

Sets the base string in the Content-Type header. The given string is copied to the Content-Type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetBase(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar                          *base);
```

PARAMETERS

hHeader

The handle of the header.

base

The base string to be set in the Content-Type header. If NULL is supplied, the existing base is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetOtherParams()

DESCRIPTION

Not all the Content-Type header parameters have separated parameters in the content-type header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function copies the Content-Type header “other-params” parameter of the content-type header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetOtherParams(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar                          *strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint                          *actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetOtherParams()

DESCRIPTION

Not all the Content-Type header parameters have separated parameters in the content-type header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function sets the “other-params” parameter in the content-type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetOtherParams(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar                          *otherParams);
```

PARAMETERS

hHeader

The handle of the header.

otherParams

The other-parameters string to be set in the Content-Type header. If NULL is supplied, the existing parameters string is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Content-Type header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetStrBadSyntax(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Content-Type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetStrBadSyntax(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvChar*                         strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetCompactForm(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvBool                          bIsCompact);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the Content Type header. This flag tells the Stack whether or not to use the compact form Content-Type header during the header encoding.

SYNTAX

```
RvStatus RvSipPartyHeaderGetCompactForm(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Party header.

pbIsCompact

Specifies whether or not to use compact form.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderGetStart()

DESCRIPTION

Retrieves the “start” parameter of the address type from the Content-Type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderGetStart(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    OUT RvSipAddressHandle              *phStartAddr);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

phStartAddr

A pointer to an address handle which is the type of the “start” parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderSetStart()

DESCRIPTION

Retrieves the “start” parameter of the address type from the Content-Type header.

SYNTAX

```
RvStatus RvSipContentTypeHeaderSetStart(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvSipAddressHandle              hStartAddr);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

hStartAddr

The handle of the “start” parameter (of the address type).

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderTypeParamGetMediaType()

DESCRIPTION

Retrieves the media type enumeration value of the “type” parameter. If RVSIP_MEDIATYPE_OTHER is returned, you can receive the media type string using RvSipContentTypeHeaderGetStrMediaType().

SYNTAX

```
RvSipMediaType RvSipContentTypeHeaderTypeParamGetMediaType(
    IN RvSipContentTypeHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

hStartAddr

The handle of the “start” parameter (of the address type).

RETURN VALUES

Returns the media type of the Content-Type “type” parameter.

RvSipContentTypeHeaderTypeParamGetStrMediaType()

DESCRIPTION

Copies the media type string from the “type” parameter of the Content-Type header into a given buffer. Use this function when the media type enumeration that RvSipContentTypeHeaderTypeParamGetMediaType() returns is RVSIP_MEDIATYPE_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderTypeParamGetStrMediaType(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderTypeParamGetMediaSubType( )

DESCRIPTION

Gets the media sub-type enumeration value of the “type” parameter. If RVSIP_MEDIASUBTYPE_OTHER is returned, you can receive the media sub-type string using RvSipContentTypeHeaderTypeParamGetStrMediaSubType().

SYNTAX

```
RvSipMediaSubType
RvSipContentTypeHeaderTypeParamGetMediaSubType(
    IN RvSipContentTypeHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

RETURN VALUES

Returns the sub-media type of the Content-Type “type” parameter.

RvSipContentTypeHeaderTypeParamGetStrMediaSubType()

DESCRIPTION

Copies the media subtype string from the “type” parameter of the Content-Type header into a given buffer. Use this function when the media sub-type enumeration that RvSipContentTypeHeaderTypeParamSetMediaSubType() returns is RVSIP_MEDIASUBTYPE_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipContentTypeHeaderTypeParamGetStrMediaSubType(
    IN  RvSipContentTypeHeaderHandle    hHeader,
    IN  RvChar*                         strBuffer,
    IN  RvUint                          bufferLen,
    OUT RvUint*                         actualLen);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderTypeParamSetMediaType()

DESCRIPTION

Sets the media type field in the “type” parameter of the Content-Type header object. If the enumeration given by *eMediaType* is RVSIP_MEDIATYPE_OTHER, sets the media type string given by *strMediaType* in the Content-Type header object. Use RVSIP_MEDIATYPE_OTHER when the media type you wish to set to the Content-Type does not have a matching enumeration value in the RvSipMediaType enumeration.

SYNTAX

```
RvStatus RvSipContentTypeHeaderTypeParamSetMediaType(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvSipMediaType                  eMediaType,
    IN RvChar                          *strMediaType);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

eMediaType

The media type enumeration to be set in the Content-Type header object. If RVSIP_MEDIATYPE_UNDEFINED is supplied, the existing media type is removed from the header.

strMediaType

The media type string to be set in the Content-Type header. (relevant when *eMediaType* is RVSIP_MEDIATYPE_OTHER).

RETURN VALUES

Returns RvStatus.

RvSipContentTypeHeaderTypeParamSetMediaSubType()

DESCRIPTION

Sets the media sub-type field in the “type” parameter of the Content-Type header object. If the enumeration given by *eMediaSubType* is RVSIP_MEDIASUBTYPE_OTHER, sets the media sub-type string that *strMediaSubType* gives in the Content-Type header object. Use RVSIP_MEDIASUBTYPE_OTHER when the media type you wish to set to the Content-Type does not have a matching enumeration value in the RvSipMediaSubType enumeration.

SYNTAX

```
RvStatus RvSipContentTypeHeaderTypeParamSetMediaSubType(
    IN RvSipContentTypeHeaderHandle    hHeader,
    IN RvSipMediaSubType               eMediaSubType,
    IN RvChar                          *strMediaSubType);
```

PARAMETERS

hHeader

The handle of the Content-Type header.

eMediaSubType

The media sub-type enumeration to be set in the Content-Type header object. If RVSIP_MEDIASUBTYPE_UNDEFINED is supplied, the existing media sub type is removed from the header.

strMediaSubType

The media sub-type string to be set in the Content-Type header. (relevant when *eMediaSubType* is RVSIP_MEDIASUBTYPE_OTHER).

RETURN VALUES

Returns RvStatus.

11

CONTENT-ID HEADER FUNCTIONS

Content-ID header functions enable you to construct, copy, encode, parse, access and change Content-ID header parameters. This section contains Content-ID header functions found in the *RvSipContentIDHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipContentIDHeaderConstruct()
- RvSipContentIDHeaderCopy()
- RvSipContentIDHeaderEncode()
- RvSipContentIDHeaderParse()
- RvSipContentIDHeaderParseValue()
- RvSipContentIDHeaderFix()

RvSipContentIDHeaderConstructInBody()

DESCRIPTION

Constructs a Content-ID header inside a given message body object.

SYNTAX

```
RvStatus RvSipContentIDHeaderConstructInBody(
IN  RvSipBodyHandle                hBody,
OUT RvSipContentIDHeaderHandle     *phHeader);
```

PARAMETERS

hBody

The handle of the message body object.

phHeader

The handle of the newly constructed Content-ID header.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Content-ID header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipContentIDHeaderConstruct(
    IN  RvSipMsgMgrHandle             hMsgMgr,
    IN  HRPOOL                        hPool,
    IN  HPAGE                         hPage,
    OUT RvSipContentIDHeaderHandle    *phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

phHeader

The handle of the newly constructed Content-ID header.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderCopy()

DESCRIPTION

Copies all fields from a source Content-ID header to a destination Content-ID header. You must construct the destination header object before using this function.

SYNTAX

```
RvStatus RvSipContentIDHeaderCopy(
    INOUT RvSipContentIDHeaderHandle    hDestination,
    IN    RvSipContentIDHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Content-ID header.

hSource

The handle of the source Content-ID header.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderEncode()

DESCRIPTION

Encodes a Content-ID header to a textual Content-ID header. The textual header is placed on a page taken from a specified pool.To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipContentIDHeaderEncode(
    IN  RvSipContentIDHeaderHandle    hHeader,
    IN  HRPOOL                        hPool,
    OUT HPAGE*                        hPage,
    OUT RvUint32*                     length);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderParse()

DESCRIPTION

Parses a SIP textual Content-ID header. For example, “Content-ID: multipart/mixed; boundart=unique”. All the textual fields are placed inside the object.

Note Before performing the parse operation the stack resets all the header fields. Therefore, if the parse function fails, you will not be able to get the previous header field values.

SYNTAX

```
RvStatus RvSipContentIDHeaderParse(
    IN RvSipContentIDHeaderHandle    hHeader,
    IN RvChar*                       buffer);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

buffer

The buffer containing the textual Content-ID header.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderParseValue()

DESCRIPTION

Parses a SIP textual Content-ID header value into a Content-ID header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipContentIDHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipContentIDHeaderParseValue(
    IN RvSipContentIDHeaderHandle    hHeader,
    IN RvChar*                       buffer);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

buffer

The buffer containing a textual Content-ID header value.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderFix()

DESCRIPTION

Fixes a Content-ID header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipContentIDHeaderFix(
    IN RvSipContentIDHeaderHandle    hHeader,
    IN RvChar*                       pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipContentIDHeaderGetStringLength()
- RvSipContentIDHeaderGetAddrSpec()
- RvSipContentIDHeaderSetAddrSpec()
- RvSipContentIDHeaderGetStrBadSyntax()
- RvSipContentIDHeaderSetStrBadSyntax()

RvSipContentIDHeaderGetStringLength()

DESCRIPTION

Some of the Content-ID header parameters are kept in a string format, such as the Content-ID header parameter boundary. In order to get such a parameter from the Content-ID header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipContentIDHeaderGetStringLength(
    IN RvSipContentIDHeaderHandle        hHeader,
    IN RvSipContentIDHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipContentIDHeaderGetAddrSpec()

DESCRIPTION

The “address spec” field is held in the Content-ID header as an *address* object. This function returns the handle of the *address*.

SYNTAX

```
RvSipAddressHandle RvSipContentIDHeaderGetAddrSpec(
    IN RvSipContentIDHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

RETURN VALUES

Returns a handle of the “address spec” *address* object, or NULL if this object does not exist.

RvSipContentIDHeaderSetAddrSpec()

DESCRIPTION

Sets the “address spec” parameter in the Content-ID header.

SYNTAX

```
RvStatus RvSipContentIDHeaderSetAddrSpec(
    IN RvSipContentIDHeaderHandle    hHeader,
    IN RvSipAddressHandle            hAddrSpec);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

hAddrSpec

The handle of the “address spec” *address* object. If NULL is supplied, the existing “address spec” is removed from the Content-ID header.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Content-ID header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipContentIDHeaderGetStrBadSyntax(
    IN  RvSipContentIDHeaderHandle    hHeader,
    IN  RvChar*                       strBuffer,
    IN  RvUint                        bufferLen,
    OUT RvUint*                       actualLen);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

strBuffer

The buffer with which to fill the bad syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad syntax + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipContentIDHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Content-ID header.

SYNTAX

```
RvStatus RvSipContentIDHeaderSetStrBadSyntax(
    IN RvSipContentIDHeaderHandle    hHeader,
    IN RvChar*                       strBadSyntax);
```

PARAMETERS

hHeader

The handle of the Content-ID header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

11

CSeq Header Functions

CSeq header functions enable you to construct, copy, encode, parse, access and change CSeq header parameters. This section contains CSeq header functions found in the *RvSipCSeqHeader.h* header file.

CSeq Working Modes

The two working modes with the CSeq header utility functions are as follows:

- Signed numeric CSeq value, which means that the maximum numeric value of the CSeq is 2^31-1. Consequently, an un-initialized numeric CSeq value is represented by the (-1) value. This mode is defined in RFC 3261, section 8.1.1.5, and thus is the default Stack mode.
- Unsigned numeric CSeq value, which means that the maximum numeric value of the CSeq is 2^32-1. This mode is defined in RFC 3261, section 20.16. To work in this mode, the RV_SIP_UNSIGNED_CSEQ compilation flag must be set.

Note Turning on the RV_SIP_UNSIGNED_CSEQ compilation flag changes the prototype definition of the RvSipCSeqHeaderGetStep() function.

The functions included in this section are:

- Control Functions

- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipCSeqHeaderConstructInMsg()
- RvSipCSeqHeaderConstruct()
- RvSipCSeqHeaderConstructInRAckHeader()
- RvSipCSeqHeaderCopy()
- RvSipCSeqHeaderEncode()
- RvSipCSeqHeaderParse()
- RvSipCSeqHeaderParseValue()
- RvSipCSeqHeaderFix()

RvSipCSeqHeaderConstructInMsg()

DESCRIPTION

Constructs a CSeq header inside a given *message*.

SYNTAX

```
RvStatus RvSipCSeqHeaderConstructInMsg(
    IN  RvSipMsgHandle           hSipMsg,
    OUT RvSipCSeqHeaderHandle    *hHeader);
```

PARAMETERS

hSipMsg

The handle of the message related to the new header.

hHeader

The handle of the CSeq header.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone CSeq header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipCSeqHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipCSeqHeaderHandle*     hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed CSeq header.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderConstructInRAckHeader()

DESCRIPTION

Constructs a CSeq header in a given RAck header. The header handle is returned.

SYNTAX

```
RvStatus RvSipCSeqHeaderConstructInRAckHeader(
    IN  RvSipRAckHeaderHandle    hHeader,
    OUT RvSipCSeqHeaderHandle    *phCSeq)
```

PARAMETERS

hHeader

The handle of the RAck header.

phCSeq

The handle of the newly constructed CSeq object.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderCopy()

DESCRIPTION

Copies all parameters from a source CSeq header to a destination CSeq header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipCSeqHeaderCopy(
    INOUT RvSipCSeqHeaderHandle    hDestination,
    IN    RvSipCSeqHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination CSeq header.

hSource

The handle of the source CSeq header.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderEncode()

DESCRIPTION

Encodes a CSeq header to a textual CSeq header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipCSeqHeaderEncode(
    IN  RvSipCSeqHeaderHandle    hHeader,
    IN  HRPOOL                   hPool,
    OUT HPAGE*                   hPage,
    OUT RvUint32*                length);
```

PARAMETERS

hHeader

The handle of the CSeq header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipCSeqHeaderParse()

DESCRIPTION

Parses a SIP textual CSeq header into a CSeq header. All the textual parameters are placed inside the object. For example, “CSeq: 1 INVITE”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipCSeqHeaderParse(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

A handle of the CSeq header.

buffer

The buffer with which to fill the requested parameter. containing a textual CSeq header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a CSeq header before using this function.

RvSipCSeqHeaderParseValue()

DESCRIPTION

Parses a SIP textual CSeq header value into an CSeq header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipCSeqHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipCSeqHeaderParseValue(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

The handle of the CSeq header.

buffer

The buffer containing a textual CSeq header value.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderFix()

DESCRIPTION

Fixes a CSeq header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipCSeqHeaderFix(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvChar*                  pFixedBuffer);
```

Parameters

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipCSeqHeaderGetStringLength()
- RvSipCSeqHeaderGetStep()
- RvSipCSeqHeaderSetStep()
- RvSipCSeqHeaderGetMethodType()
- RvSipCSeqHeaderGetStrMethodType()
- RvSipCSeqHeaderSetMethodType()
- RvSipCSeqHeaderGetStrBadSyntax()
- RvSipCSeqHeaderSetStrBadSyntax()

RvSipCSeqHeaderGetStringLength()

DESCRIPTION

Some of the CSeq header parameters are kept in a string format, for example, the CSeq header method. In order to get such a parameter from the CSeq header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipCSeqHeaderGetStringLength(
    IN RvSipCSeqHeaderHandle        hHeader,
    IN RvSipCSeqHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the CSeq header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipCSeqHeaderGetStep()

DESCRIPTION

Gets the step value from the CSeq header. This function has two prototype definitions, according to the working mode that you choose. For more information, see CSeq Working Modes.

SIGNED CSEQ MODE

SYNTAX

```
RvInt32 RvSipCSeqHeaderGetStep(
    IN RvSipCSeqHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the CSeq header.

RETURN VALUES

Returns the step number value, or UNDEFINED if the step is not set.

UNSIGNED CSEQ MODE

SYNTAX

```
RvStatus RvSipCSeqHeaderGetStep(
    IN  RvSipCSeqHeaderHandle    hHeader,
    OUT RvUint32                 *pStep);
```

PARAMETERS

hHeader

The handle of the CSeq header.

pStep

A pointer to an unsigned variable to be filled with the CSeq step value, only if the CSeq header step is already initialized.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderSetStep()

DESCRIPTION

Sets the CSeq step value in the CSeq header.

SYNTAX

```
RvStatus RvSipCSeqHeaderSetStep(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvInt32                  step);
```

PARAMETERS

hHeader

The handle of the CSeq header.

step

CSeq value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderGetMethodType()

DESCRIPTION

Gets the method type enumeration from the CSeq header.

SYNTAX

```
RvSipMethodType RvSipCSeqHeaderGetMethodType(
    IN RvSipCSeqHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the CSeq header.

RETURN VALUES

Returns the method type enumeration from the CSeq header.

REMARKS

If this function returns RVSIP_METHOD_OTHER, call RvSipCSeqHeaderGetStrMethodType() to get the actual method in a string format.

RvSipCSeqHeaderGetStrMethodType()

DESCRIPTION

Copies the method type string from the CSeq header into a given buffer. Use this function if RvSipCSeqHeaderGetMethodType() returns RVSIP_METHOD_OTHER. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipCSeqHeaderGetStrMethodType(
    IN  RvSipCSeqHeaderHandle    hHeader,
    IN  RvChar*                  strBuffer,
    IN  RvUint                   bufferLen,
    OUT RvUint*                  actualLen);
```

PARAMETERS

hHeader

The handle of the CSeq header.

strBuffer

The buffer with which to fill the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderSetMethodType()

DESCRIPTION

Sets the method type in the CSeq object.

SYNTAX

```
RvStatus RvSipCSeqHeaderSetMethodType(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvSipMethodType          eMethodType,
    IN RvChar*                  strMethodType);
```

PARAMETERS

hHeader

The handle of the CSeq header.

eMethodType

The method type to be set in the object.

strMethodType

You can use this parameter only if the *eMethodType* parameter is set to RVSIP_METHOD_OTHER. In this case, you can supply the method as a string.

RETURN VALUES

Returns RvStatus.

REMARKS

If eMethodType is RVSIP_METHOD_OTHER, *strMethodType* is copied to the header. Otherwise *strMethodType* is ignored.

RvSipCSeqHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad CSeq header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipCSeqHeaderGetStrBadSyntax(
    IN  RvSipCSeqHeaderHandle    hHeader,
    IN  RvChar*                  strBuffer,
    IN  RvUint                   bufferLen,
    OUT RvUint*                  actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipCSeqHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal CSeq header.

SYNTAX

```
RvStatus RvSipCSeqHeaderSetStrBadSyntax(
    IN RvSipCSeqHeaderHandle    hHeader,
    IN RvChar*                  strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipCSeqHeaderSetStrBadSyntax()

12

DATE HEADER FUNCTIONS

The Date header functions enable you to construct, destruct, copy, encode, parse, access and change Date header parameters. This section contains Date header functions found in the *RvSipDateHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipDateHeaderConstructInMsg()
- RvSipDateConstructInExpiresHeader()
- RvSipDateConstructInRetryAfterHeader()
- RvSipDateHeaderConstruct()
- RvSipDateHeaderCopy()
- RvSipDateHeaderEncode()
- RvSipDateHeaderParse()
- RvSipDateHeaderParseValue()
- RvSipDateHeaderFix()

RvSipDateHeaderConstructInMsg()

DESCRIPTION

Constructs a Date header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipDateHeaderConstructInMsg(
    IN  RvSipMsgHandle            hSipMsg,
    IN  RvBool                    pushHeaderAtHead,
    OUT RvSipDateHeaderHandle*    phHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Date header.

RETURN VALUES

Returns RvStatus.

RvSipDateConstructInExpiresHeader()

DESCRIPTION

Constructs a Date header in a given Expires header. The header handle is returned.

SYNTAX

```
RvStatus RvSipDateConstructInExpiresHeader(
    IN  RvSipExpiresHeaderHandle    hHeader,
    OUT RvSipDateHeaderHandle       *phDate);
```

PARAMETERS

hHeader

The handle of the Expires header that relates to this date.

phDate

The handle of the newly constructed date object.

RETURN VALUES

Returns RvStatus.

RvSipDateConstructInRetryAfterHeader()

DESCRIPTION

Constructs a Date header in a given Retry-After header. The header handle is returned.

SYNTAX

```
RvStatus RvSipDateConstructInRetryAfterHeader(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    OUT RvSipDateHeaderHandle          *phDate);
```

PARAMETERS

hHeader

The handle of the Retry-After header that relates to this date.

phDate

The handle of the newly constructed date object.

RETURN VALUES

Returns RV_OK, RV_ERROR_INVALID_HANDLE, RV_ERROR_NULLPTR, or RV_ERROR_OUTOFRESOURCES.

RvSipDateHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Date header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipDateHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipDateHeaderHandle*     phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Date header.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderCopy()

DESCRIPTION

Copies all parameters from a source Date header to a destination Date header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipDateHeaderCopy(
    INOUT RvSipDateHeaderHandle    hDestination,
    IN    RvSipDateHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Date header.

hSource

The handle of the source Date header.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderEncode()

DESCRIPTION

Encodes a Date header to a textual Date header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipDateHeaderEncode(
    IN  RvSipDateHeaderHandle    hHeader,
    IN  HRPOOL                   hPool,
    OUT HPAGE*                   phPage,
    OUT RvUint32*                pLength);
```

PARAMETERS

hHeader

The handle of the Date header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

pLength

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipDateHeaderParse()

DESCRIPTION

Parses a SIP textual Date header into a Date header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipDateHeaderParse(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvChar*                  strBuffer);
```

PARAMETERS

hHeader

A handle of the Date header.

buffer

The buffer containing a textual Date header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Date header before using this function.

RvSipDateHeaderParseValue()

DESCRIPTION

Parses a SIP textual Date header value into an Date header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipDateHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipDateHeaderParseValue(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

The handle of the Date header.

buffer

The buffer containing a textual Date header value.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderFix()

DESCRIPTION

Fixes a Date header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipDateHeaderFix(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvChar*                  pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipDateHeaderGetWeekDay()
- RvSipDateHeaderSetWeekDay()
- RvSipDateHeaderGetDay()
- RvSipDateHeaderSetDay()
- RvSipDateHeaderGetMonth()
- RvSipDateHeaderSetMonth()
- RvSipDateHeaderGetYear()
- RvSipDateHeaderSetYear()
- RvSipDateHeaderGetHour()
- RvSipDateHeaderSetHour()
- RvSipDateHeaderGetMinute()
- RvSipDateHeaderSetMinute()
- RvSipDateHeaderGetSecond()
- RvSipDateHeaderSetSecond()
- RvSipDateHeaderGetStringLength()
- RvSipDateHeaderGetStrBadSyntax()
- RvSipDateHeaderSetStrBadSyntax()

RvSipDateHeaderGetWeekDay()

DESCRIPTION

Gets the week-day enumeration. RVSIP_WEEKDAY_UNDEFINED is returned if the weekday is not set.

SYNTAX

```
RvSipDateWeekDay RvSipDateHeaderGetWeekDay(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

The enumeration of the week day.

RvSipDateHeaderSetWeekDay()

DESCRIPTION

Sets the week-day parameter in the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetWeekDay(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvSipDateWeekDay         eWeekDay);
```

PARAMETERS

hHeader

The handle of the Date header.

eWeekDay

The week-day enumeration.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetDay()

DESCRIPTION

Gets the number for the day of the month in the given Date header.

SYNTAX

```
RvInt8 RvSipDateHeaderGetDay(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the day number. If the integer number for the day of the month is not set, UNDEFINED is returned.

RvSipDateHeaderSetDay()

DESCRIPTION

Sets the number for the day of the month parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetDay(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvInt8                   day);
```

PARAMETERS

hHeader

The handle of the Date header.

day

The day number.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetMonth()

DESCRIPTION

Gets the month enumeration of the given Date header.
RVSIP_MONTH_UNDEFINED is returned if the month parameter is not set.

SYNTAX

```
RvSipDateMonth RvSipDateHeaderGetMonth(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the month enumeration.

RvSipDateHeaderSetMonth()

DESCRIPTION

Sets the month parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetMonth(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvSipDateMonth           eMonth);
```

PARAMETERS

hHeader

The handle of the Date header.

eMonth

The month enumeration.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetYear()

DESCRIPTION

Gets the year of the given Date header.

SYNTAX

```
RvInt16 RvSipDateHeaderGetYear(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the year number. If the year parameter is not set, UNDEFINED is returned.

RvSipDateHeaderSetYear()

DESCRIPTION

Sets the year number parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetYear(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvInt16                  year);
```

PARAMETERS

hHeader

The handle of the Date header.

year

The year number.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetHour()

DESCRIPTION

Gets the hour of the given Date header.

SYNTAX

```
RvInt8 RvSipDateHeaderGetHour(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the hour number. If the hour parameter is not set, UNDEFINED is returned.

RvSipDateHeaderSetHour()

DESCRIPTION

Sets the hour number parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetHour(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvInt8                   hour);
```

PARAMETERS

hHeader

The handle of the Date header.

hour

The hour number.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetMinute()

DESCRIPTION

Gets the minute of the given Date header.

SYNTAX

```
RvInt8 RvSipDateHeaderGetMinute(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the minute number. If the minute parameter is not set, UNDEFINED is returned.

RvSipDateHeaderSetMinute()

DESCRIPTION

Sets the minute number parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetMinute(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvInt8                   minute);
```

PARAMETERS

hHeader

The handle of the Date header.

minute

The minute number.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetSecond()

DESCRIPTION

Gets the second number of the given Date header.

SYNTAX

```
RvInt8 RvSipDateHeaderGetSecond(
    IN RvSipDateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Date header.

RETURN VALUES

Returns the second number. If the second parameter is not set, UNDEFINED is returned.

RvSipDateHeaderSetSecond()

DESCRIPTION

Sets the second number parameter of the Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetSecond(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvInt8                   second);
```

PARAMETERS

hHeader

The handle of the Date header.

second

The second number.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetStringLength()

DESCRIPTION

Some of the Date header parameters are kept in a string format. In order to get such a parameter from the Date header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipDateHeaderGetStringLength(
    IN RvSipDateHeaderHandle        hHeader,
    IN RvSipDateHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Date header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name: header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Date header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipDateHeaderGetStrBadSyntax(
    IN  RvSipDateHeaderHandle    hHeader,
    IN  RvChar*                  strBuffer,
    IN  RvUint                   bufferLen,
    OUT RvUint*                  actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipDateHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Date header.

SYNTAX

```
RvStatus RvSipDateHeaderSetStrBadSyntax(
    IN RvSipDateHeaderHandle    hHeader,
    IN RvChar                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

13

EVENT HEADER FUNCTIONS

The Event header functions enable you to construct, copy, encode, parse, access and change Event header parameters. This section contains Event header functions found in the *RvSipEventHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipEventHeaderConstructInMsg()
- RvSipEventHeaderConstruct()
- RvSipEventHeaderCopy()
- RvSipEventHeaderEncode()
- RvSipEventHeaderParse()
- RvSipEventHeaderParseValue()
- RvSipEventHeaderFix()
- RvSipEventHeaderIsEqual()

RvSipEventHeaderConstructInMsg()

DESCRIPTION

Constructs an Event header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipEventHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    IN  RvBool                     pushHeaderAtHead,
    OUT RvSipEventHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE), or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Event header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipEventHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipEventHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderCopy()

DESCRIPTION

Copies all parameters from a source Event header to a destination Event header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipEventHeaderCopy(
    INOUT RvSipEventHeaderHandle    hDestination,
    IN    RvSipEventHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Event header.

hSource

The handle of the source Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderEncode()

DESCRIPTION

Encodes an Event header to a textual Event header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatusRvSipEventHeaderEncode(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  HRPOOL                    hPool,
    OUT HPAGE*                    hPage,
    OUT RvUint32*                 length);
```

PARAMETERS

hHeader

The handle of the Event header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderParse()

DESCRIPTION

Parses a SIP textual Event header into an Event header. An example of a SIP textual Event header is:

“Event: event-package.event-template;id=5;event-param=param-val”.

All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipEventHeaderParse(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

A handle of the Event header.

buffer

The buffer containing a textual Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderParseValue()

DESCRIPTION

Parses a SIP textual Event header value into an Event header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipEventHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipEventHeaderParseValue(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

The handle of the Event header.

buffer

The buffer containing a textual Event header value.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderFix()

DESCRIPTION

Fixes an Event header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipEventHeaderFix(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar*                   pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderIsEqual()

DESCRIPTION

Compares two Event headers. Event headers are considered equal if the event-type and event ID parameter are equal. The event-type portion of the Event header is compared byte-by-byte, and the ID parameter token (if present) is compared byte-by-byte. An Event header containing an ID parameter never matches an Event header without an ID parameter. No other-parameters are considered when performing a comparison.

SYNTAX

```
RvBool RvSipEventHeaderIsEqual(
    IN const RvSipEventHeaderHandle    hHeader,
    IN const RvSipEventHeaderHandle    hOtherHeader);
```

PARAMETERS

hHeader

The handle of the Event header.

hOtherHeader

The handle of the Event header with which a comparison is being made.

RETURN VALUES

Returns RV_TRUE if the Party headers being compared are equal. Otherwise returns RV_FALSE.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipEventHeaderGetStringLength()
- RvSipEventHeaderGetRpoolString()
- RvSipEventHeaderSetRpoolString()
- RvSipEventHeaderGetEventPackage()
- RvSipEventHeaderSetEventPackage()
- RvSipEventHeaderGetEventTemplate()
- RvSipEventHeaderSetEventTemplate()
- RvSipEventHeaderGetEventParam()
- RvSipEventHeaderSetEventParam()
- RvSipEventHeaderGetEventId()
- RvSipEventHeaderSetEventId()
- RvSipEventHeaderGetStrBadSyntax()
- RvSipEventHeaderSetStrBadSyntax()
- RvSipEventHeaderSetCompactForm()
- RvSipEventHeaderGetCompactForm()

RvSipEventHeaderGetStringLength()

DESCRIPTION

Some of the Event header parameters are kept in a string format, for example, the Event header “other-params”. In order to get such a parameter from the Event header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipEventHeaderGetStringLength(
    IN RvSipEventHeaderHandle        hHeader,
    IN RvSipEventHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Event header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipEventHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Event header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipEventHeaderGetRpoolString(
    IN    RvSipEventHeaderHandle        hSipEventHeader,
    IN    RvSipEventHeaderStringName    eStringName,
    INOUT RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipEventHeader

The handle of the Event header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Via header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Event header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipEventHeaderSetRpoolString(
    IN RvSipEventHeaderHandle        hSipEventHeader,
    IN RvSipEventHeaderStringName    eStringName,
    IN RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipEventHeader

The handle of the Event header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetEventPackage()

DESCRIPTION

Copies the event-package string from the Event header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipEventHeaderGetEventPackage(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetEventPackage()

DESCRIPTION

Sets the event-package parameter in the Event header.

SYNTAX

```
RvStatus RvSipEventHeaderSetEventPackage(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar                    *strPackage);
```

PARAMETERS

hHeader

The handle of the Event header.

strPackage

The event-package string to be set in the Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetEventTemplate()

DESCRIPTION

Copies the event-template string from the Event header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipEventHeaderGetEventTemplate(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter+ 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetEventTemplate()

DESCRIPTION

Sets the event-template parameter in the Event header.

SYNTAX

```
RvStatus RvSipEventHeaderSetEventTemplate(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar                    *strTemplate);
```

PARAMETERS

hHeader

The handle of the Event header.

strPackage

The event-template string to be set in the Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetEventParam()

DESCRIPTION

Copies the event-param string from the Event header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipEventHeaderGetEventParam(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetEventParam()

DESCRIPTION

Sets the event-param parameter in the Event header.

SYNTAX

```
RvStatus RvSipEventHeaderSetEventParam(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar                    strEventParam);
```

PARAMETERS

hHeader

The handle of the Event header.

strEventParam

The event-param string to be set in the Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetEventId()

DESCRIPTION

Copies the event-id string from the Event header into a given buffer. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipEventHeaderGetEventId(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetEventId()

DESCRIPTION

Sets the event-id parameter in the Event header.

SYNTAX

```
RvStatus RvSipEventHeaderSetEventId(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar                    *strId);
```

PARAMETERS

hHeader

The handle of the Event header.

strId

The event-id string to be set in the Event header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Event header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipEventHeaderGetStrBadSyntax(
    IN  RvSipEventHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Event header.

SYNTAX

```
RvStatus RvSipEventHeaderSetStrBadSyntax(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvChar*                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipEventHeaderSetCompactForm(
    IN RvSipEventHeaderHandle    hHeader,
    IN RvBool                    bIsCompact);
```

PARAMETERS

hHeader

The handle of the Event header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipEventHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the Event header.

SYNTAX

```
RvStatus RvSipEventHeaderGetCompactForm (
    IN RvSipEventHeaderHandle    hHeader,
    IN RvBool                    *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Event header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipEventHeaderGetCompactForm()

14

EXPIRES HEADER FUNCTIONS

The Expires header functions enable you to construct, copy, encode, parse, access and change Expires header parameters. This section contains Expires header functions found in the *RvSipExpiresHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipExpiresHeaderConstructInMsg()
- RvSipExpiresConstructInContactHeader()
- RvSipExpiresHeaderConstruct()
- RvSipExpiresHeaderCopy()
- RvSipExpiresHeaderEncode()
- RvSipExpiresHeaderParse()
- RvSipExpiresHeaderParseValue()
- RvSipExpiresHeaderFix()

RvSipExpiresHeaderConstructInMsg()

DESCRIPTION

Constructs a Expires header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipExpiresHeaderConstructInMsg(
    IN  RvSipMsgHandle               hSipMsg,
    IN  RvBool                       pushHeaderAtHead,
    OUT RvSipExpiresHeaderHandle*    phHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Expires header.

RETURN VALUES

Returns RvStatus.

RvSipExpiresConstructInContactHeader()

DESCRIPTION

Constructs an expires object inside a given Contact header. The header handle is returned.

SYNTAX

```
RvStatus RvSipExpiresConstructInContactHeader(
    IN  RvSipContactHeaderHandle    hHeader,
    OUT RvSipExpiresHeaderHandle    *phExpires);
```

PARAMETERS

hHeader

The handle of the Contact header that relates to this expires object.

phExpires

The handle of the newly constructed expires object.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Expires header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipExpiresHeaderConstruct(
    IN  RvSipMsgMgrHandle            hMsgMgr,
    IN  HRPOOL                       hPool,
    IN  HPAGE                        hPage,
    OUT RvSipExpiresHeaderHandle*    phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Expires header.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderCopy()

DESCRIPTION

Copies all parameters from a source Expires header to a destination Expires header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipExpiresHeaderCopy(
    INOUT RvSipExpiresHeaderHandle    hDestination,
    IN    RvSipExpiresHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Expires header.

hSource

The handle of the source Expires header.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderEncode()

DESCRIPTION

Encodes a Expires header to a textual Expires header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipExpiresHeaderEncode(
    IN  RvSipExpiresHeaderHandle    hHeader,
    IN  HRPOOL                      hPool,
    OUT HPAGE*                      phPage,
    OUT RvUint32*                   pLength);
```

PARAMETERS

hHeader

The handle of the Expires header.

hPool

The handle of the specified memory pool.

phPage

The memory page allocated to contain the encoded header.

plength

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipExpiresHeaderParse()

DESCRIPTION

Parses a SIP textual Expires header into a Expires header. All the textual parameters are placed inside the object. For example, “Expires: Thu, 01 Dec 2040 16:00:00 GMT”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipExpiresHeaderParse(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvChar*                     strBuffer);
```

PARAMETERS

hHeader

A handle of the Expires header.

buffer

The buffer containing a textual Expires header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Expires header before using this function.

RvSipExpiresHeaderParseValue()

DESCRIPTION

Parses a SIP textual Expires header value into an Expires header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipExpiresHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipExpiresHeaderParseValue(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvChar*                     buffer);
```

PARAMETERS

hHeader

The handle of the Expires header.

buffer

The buffer containing a textual Expires header value.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderFix()

DESCRIPTION

Fixes an Expires header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipExpiresHeaderFix(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvChar*                     pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipExpiresHeaderGetFormat()
- RvSipExpiresHeaderGetDeltaSeconds()
- RvSipExpiresHeaderSetDeltaSeconds()
- RvSipExpiresHeaderGetDateHandle()
- RvSipExpiresHeaderSetDateHandle()
- RvSipExpiresHeaderGetStringLength()
- RvSipExpiresHeaderGetStrBadSyntax()
- RvSipExpiresHeaderSetStrBadSyntax()

RvSipExpiresHeaderGetFormat()

DESCRIPTION

Gets the format of the Expires header: Undefined, Date or Delta-seconds.

SYNTAX

```
RvSipExpiresFormat RvSipExpiresHeaderGetFormat(
    IN RvSipExpiresHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Expires header.

RETURN VALUES

The format enumeration.

RvSipExpiresHeaderGetDeltaSeconds()

DESCRIPTION

Gets the delta-seconds integer of the Expires header. If the delta-seconds integer is not set, UNDEFINED is returned.

SYNTAX

```
RvStatus RvSipExpiresHeaderGetDeltaSeconds(
    IN  RvSipExpiresHeaderHandle    hHeader,
    OUT RvUint32                    *pDeltaSeconds);
```

PARAMETERS

hHeader

The handle of the Expires header.

pDeltaSeconds

The delta-seconds integer.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderSetDeltaSeconds()

DESCRIPTION

Sets the delta seconds integer of the Expires header. Changes the Expires format to delta-seconds. If the given delta-seconds is UNDEFINED, the delta-seconds of the Expires header is removed and the format is changed to UNDEFINED.

SYNTAX

```
RvStatus RvSipExpiresHeaderSetDeltaSeconds(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvUint32                    deltaSeconds);
```

PARAMETERS

hHeader

The handle of the Expires header.

deltaSeconds

The delta-seconds to be set to the Expires header.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderGetDateHandle()

DESCRIPTION

Gets the date handle of the Expires header.

SYNTAX

```
RvSipDateHeaderHandle RvSipExpiresHeaderGetDateHandle(
    IN RvSipExpiresHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Expires header.

RETURN VALUES

Returns the handle of the Date header, or NULL if the Date header does not exist.

RvSipExpiresHeaderSetDateHandle()

DESCRIPTION

Sets a new Date header in the Expires header and changes the Expires format to date.

SYNTAX

```
RvStatus RvSipExpiresHeaderSetDateHandle(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvSipDateHeaderHandle       hDate);
```

PARAMETERS

hHeader

The handle of the Expires header.

hDate

The date handle of be set to the Expires header. If the date handle is NULL, the existing Date header is removed from the Expires header.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderGetStringLength()

DESCRIPTION

Some of the Expires header parameters are kept in a string format. In order to get such a parameter from the Expires header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to let you allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipExpiresHeaderGetStringLength(
    IN RvSipExpiresHeaderHandle        hHeader,
    IN RvSipExpiresHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Expires header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipExpiresHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Expires header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipExpiresHeaderGetStrBadSyntax(
    IN  RvSipExpiresHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipExpiresHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Expires header.

SYNTAX

```
RvStatus RvSipExpiresHeaderSetStrBadSyntax(
    IN RvSipExpiresHeaderHandle    hHeader,
    IN RvChar*                     strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

15

MIN-SE HEADER FUNCTIONS

The Min-SE header functions enable you to construct, copy, encode, parse, access and change Min-SE header parameters. This section contains Min-SE header functions found in the *RvSipMin-SEHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipMinSEHeaderConstructInMsg()
- RvSipMinSEHeaderConstruct()
- RvSipMinSEHeaderCopy()
- RvSipMinSEHeaderEncode()
- RvSipMinSEHeaderParse()
- RvSipMinSEHeaderParseValue()
- RvSipMinSEHeaderFix()

RvSipMinSEHeaderConstructInMsg()

DESCRIPTION

Constructs a Min-SE header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipMinSEHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    IN  RvBool                     pushHeaderAtHead,
    OUT RvSipMinSEHeaderHandle*    phHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

phHeader

The handle of the newly constructed Min-SE header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Min-SE header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipMinSEHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipMinSEHeaderHandle*    phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

phHeader

The handle of the newly constructed Min-SE header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderCopy()

DESCRIPTION

Copies all parameters from a source Min-SE header to a destination Min-SE header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipMinSEHeaderCopy(
    INOUT    RvSipMinSEHeaderHandle    hDestination,
    IN RvSipMinSEHeaderHandle          hSource);
```

PARAMETERS

pDestination

The handle of the destination Min-SE header.

pSource

The handle of the source Min-SE header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderEncode()

DESCRIPTION

Encodes a Min-SE header to a textual Min-SE header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal(). The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

SYNTAX

```
RvStatus RvSipMinSEHeaderEncode(
    IN  RvSipMinSEHeaderHandle    hHeader,
    IN  HRPOOL                    hPool,
    OUT HPAGE*                    phPage,
    OUT RvUint32*                 pLength);
```

PARAMETERS

hHeader

The handle of the Min-SE header.

hPool

The handle of the specified memory pool.

phPage

The memory page allocated to contain the encoded header.

pLength

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderParse()

DESCRIPTION

Parses a SIP textual Min-SE header into a Min-SE header. An example of a SIP textual Min-SE header is “Min-SE: 3600”. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipMinSEHeaderParse(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvChar*                   strBuffer);
```

PARAMETERS

hHeader

A handle of the Min-SE header.

strBuffer

The buffer containing a textual Min-SE header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderParseValue()

DESCRIPTION

Parses a SIP textual Min-SE header value into an Min-SE header. A SIP header has the following grammer: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipMinSEHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipMinSEHeaderParseValue(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

The handle of the Min-SE header.

buffer

The buffer containing a textual Min-SE header value.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderFix()

DESCRIPTION

Fixes a Min-SE header with bad-syntax information. A SIP header has the following grammer: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipMinSEHeaderFix(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvChar*                   pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipMinSEHeaderGetDeltaSeconds()
- RvSipMinSEHeaderSetDeltaSeconds()
- RvSipMinSEHeaderGetOtherParams()
- RvSipMinSEHeaderSetOtherParams()
- RvSipMinSEHeaderGetStringLength()
- RvSipMinSEHeaderGetStrBadSyntax()
- RvSipMinSEHeaderSetStrBadSyntax()

RvSipMinSEHeaderGetDeltaSeconds()

DESCRIPTION

Gets the delta-seconds integer of the Min-SE header. If the delta-seconds integer is not set, UNDEFINED is returned.

SYNTAX

```
RvStatus RvSipMinSEHeaderGetDeltaSeconds(
    IN  RvSipMinSEHeaderHandle    hHeader,
    OUT RvInt32                   *pDeltaSeconds);
```

PARAMETERS

hHeader

The handle of the Min-SE header.

pDeltaSeconds

The delta-seconds integer.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderSetDeltaSeconds()

DESCRIPTION

Sets the delta seconds integer of the Min-SE header. If the given delta-seconds is UNDEFINED, the delta-seconds of the Min-SE header is removed

SYNTAX

```
RvStatus RvSipMinSEHeaderSetDeltaSeconds(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvInt32                   deltaSeconds);
```

PARAMETERS

hHeader

The handle of the Min-SE header.

deltaSeconds

The delta-seconds to be set to the Min-SE header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderGetOtherParams()

DESCRIPTION

Copies the other-params string from the Min-SE header into a given buffer. Not all the Content-Type header parameters have separated parameters in the Content-Type header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipMinSEHeaderGetOtherParams(
    IN  RvSipMinSEHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderSetOtherParams()

DESCRIPTION

Sets the other-params string in the Min-SE header. The given string is copied to the Min-SE header. Not all the Min-SE header parameters have separated parameters in the Min-SE header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipMinSEHeaderSetOtherParams(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvChar                    *otherParams);
```

PARAMETERS

hHeader

The handle of the header.

otherParams

The other-parameters string to be set in the Min-SE header. If NULL is supplied, the existing parameters string is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderGetStringLength()

DESCRIPTION

The other-parameters of Min-SE header parameters are kept in a string format. In order to get such a parameter from the Min-SE header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enables you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipMinSEHeaderGetStringLength(
    IN RvSipMinSEHeaderHandle        hHeader,
    IN RvSipMinSEHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Min-SE header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipMinSEHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammer: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback if you wish to see the header-value.

SYNTAX

```
RvStatus RvSipMinSEHeaderGetStrBadSyntax(
    IN  RvSipMinSEHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipMinSEHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammer: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Min-SE header.

SYNTAX

```
RvStatus RvSipMinSEHeaderSetStrBadSyntax(
    IN RvSipMinSEHeaderHandle    hHeader,
    IN RvChar*                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

17

OTHER HEADER FUNCTIONS

Other header functions enable you to construct, copy, encode, parse, access and change Other header parameters. Other headers should be used for every header that has no specific object in the message API. for example, a supported header can be held in an Other header. This section contains the Other header functions included in the *RvSipOtherHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipOtherHeaderConstructInMsg()
- RvSipOtherHeaderConstructInBodyPart()
- RvSipOtherHeaderConstruct()
- RvSipOtherHeaderCopy()
- RvSipOtherHeaderEncode()
- RvSipOtherHeaderParse()

RvSipOtherHeaderConstructInMsg()

DESCRIPTION

Constructs an Other header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipOtherHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    IN  RvBool                     pushHeaderAtHead,
    OUT RvSipOtherHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Other header.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderConstructInBodyPart()

DESCRIPTION

Constructs an Other header inside a given message body part object. The header is kept in the Other headers list of the message body part object. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipOtherHeaderConstructInBodyPart(
    IN  RvSipBodyPartHandle        hBodyPart,
    IN  RvBool                     pushHeaderAtHead,
    OUT RvSipOtherHeaderHandle*    hHeader);
```

PARAMETERS

hBodyPart

The handle of the message body-part object.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Other header.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Other header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipOtherHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipOtherHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Other header.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderCopy()

DESCRIPTION

Copies all parameters from a source Other header to a destination Other header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipOtherHeaderCopy(
    INOUT RvSipOtherHeaderHandle    hDestination,
    IN    RvSipOtherHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Other header.

hSource

The handle of the source Other header.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderEncode()

DESCRIPTION

Encodes a Contact header to a textual Other header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipOtherHeaderEncode(
    IN  RvSipOtherHeaderHandle    hHeader,
    IN  HRPOOL                    hPool,
    OUT HPAGE*                    hPage,
    OUT RvUint32*                 length);
```

PARAMETERS

hHeader

The handle of the Other header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipOtherHeaderParse()

DESCRIPTION

Parses a SIP textual Other header into a Other header. All the textual parameters are placed inside the object. For example, “Content–Disposition: session”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipOtherHeaderParse(
    INOUT RvSipOtherHeaderHandle    hHeader,
    IN    RvChar*                   buffer);
```

PARAMETERS

hHeader

A handle of the Other header.

buffer

The buffer containing a textual Other header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Other header before using this function.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipOtherHeaderGetStringLength()
- RvSipOtherHeaderGetName()
- RvSipOtherHeaderSetName()
- RvSipOtherHeaderGetValue()
- RvSipOtherHeaderSetValue()
- RvSipOtherHeaderGetRpoolString()
- RvSipOtherHeaderSetRpoolString()
- RvSipOtherHeaderGetStrBadSyntax()
- RvSipOtherHeaderSetStrBadSyntax()

RvSipOtherHeaderGetStringLength()

DESCRIPTION

Some of the Other header parameters are kept in a string format, for example, the Other header display name. In order to get such a parameter from the Other header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipOtherHeaderGetStringLength(
    IN RvSipOtherHeaderHandle        hHeader,
    IN RvSipOtherHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Other header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipOtherHeaderGetName()

DESCRIPTION

Copies the name of the Other header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipOtherHeaderGetName(
    IN  RvSipOtherHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderSetName()

DESCRIPTION

Sets the name of the header.

SYNTAX

```
RvStatus RvSipOtherHeaderSetName(
    IN RvSipOtherHeaderhandle    hHeader,
    IN RvChar*                   strName);
```

PARAMETERS

hHeader

The handle of the Other header.

strName

The name of the header to be set in the Other header. If NULL is supplied, the existing header name is removed.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderGetValue()

DESCRIPTION

Copies the value of the Other header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipOtherHeaderGetValue(
    IN  RvSipOtherHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the Other header.

strBuffer

buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderSetValue()

DESCRIPTION

Sets the value of the header.

SYNTAX

```
RvStatus RvSipOtherHeaderSetValue(
    IN RvSipOtherHeaderHandle    hHeader,
    IN RvChar*                   strValue);
```

PARAMETERS

hHeader

The handle of the Other header.

strValue

The header value to be set in the Other header. If NULL is supplied, the existing header value is removed.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Other header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipOtherHeaderGetRpoolString(
    IN    RvSipOtherHeaderHandle        hSipOtherHeader,
    IN    RvSipOtherHeaderStringName    eStringName,
    INOUT RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipOtherHeader

The handle of the Other header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Other header.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Other header. The given string is located on an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipOtherHeaderSetRpoolString(
    IN RvSipOtherHeaderHandle        hSipOtherHeader,
    IN RvSipOtherHeaderStringName    eStringName,
    IN RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipOtherHeader

The handle of the Other header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Other header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipOtherHeaderGetStrBadSyntax(
    IN  RvSipOtherHeaderHandle hHeader,
    IN  RvChar*     strBuffer,
    IN  RvUint      bufferLen,
    OUT RvUint*     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipOtherHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Other header.

SYNTAX

```
RvStatus RvSipOtherHeaderSetStrBadSyntax(
    IN RvSipOtherHeaderHandle    hHeader,
    IN RvChar*                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipOtherHeaderSetStrBadSyntax()

18

PARTY HEADER FUNCTIONS

The Party header functions enable you to construct, copy, encode, parse and access Party header parameters. This section contains Party header functions, for To/From headers, found in the *RvSipPartyHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipToHeaderConstructInMsg()
- RvSipFromHeaderConstructInMsg()
- RvSipPartyHeaderConstruct()
- RvSipPartyHeaderCopy()
- RvSipPartyHeaderEncode()
- RvSipPartyHeaderParse()
- RvSipPartyHeaderParseValue()
- RvSipPartyHeaderFix()
- RvSipPartyIsEqual()

RvSipToHeaderConstructInMsg()

DESCRIPTION

Constructs a To Party header inside a given *message*.

SYNTAX

```
RvStatus RvSipToHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    OUT RvSipPartyHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hHeader

The handle of the newly constructed To Party header.

RETURN VALUES

Returns RvStatus.

RvSipFromHeaderConstructInMsg()

DESCRIPTION

Constructs a From Party header inside a given *message*.

SYNTAX

```
RvStatus RvSipFromHeaderConstructInMsg(
    IN  RvSipMsgHandle             hSipMsg,
    OUT RvSipPartyHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

hHeader

The handle of the newly constructed From Party header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Party header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipPartyHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipPartyHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Party header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderCopy()

DESCRIPTION

Copies all parameters from a source Party header to a destination Party header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipPartyHeaderCopy(
    INOUT RvSipPartyHeaderHandle    hDestination,
    IN    RvSipPartyHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Party header.

hSource

The handle of the source Party header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderEncode()

DESCRIPTION

Encodes a Party header to a textual Party header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipPartyHeaderEncode(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  HRPOOL                    hPool,
    IN  RvBool                    bIsTo,
    OUT HPAGE*                    hPage,
    OUT RvUint32*                 length);
```

PARAMETERS

hHeader

The handle of the Contact header.

hPool

The handle of the specified memory pool.

bIsTo

Indicates whether a header is a To header (RV_TRUE) or From header (RV_FALSE).

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipPartyHeaderParse()

DESCRIPTION

Parses a SIP textual Party header into a Party header. All the textual parameters are placed inside the object. For example, “From: <sip:charlie@caller.com>;tag=5”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipPartyHeaderParse(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    bIsTo,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

A handle of the Party header.

bIsTo

Indicates whether a header is a To header (RV_TRUE) or From header (RV_FALSE).

buffer

The buffer containing a textual Party header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Party header before using this function.

RvSipPartyHeaderParseValue()

DESCRIPTION

Parses a SIP textual Party header value into an Party header. A SIP header has the following grammar “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipPartyHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatusRvSipPartyHeaderParseValue(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    bIsTo,
    IN RvChar*                   buffer);
```

PARAMETERS

hHeader

The handle of the Party header.

bIsTo

Indicates whether a header is a To header (RV_TRUE) or From header (RV_FALSE).

buffer

The buffer containing a textual Party header value.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderFix()

DESCRIPTION

Fixes a Party header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipPartyHeaderFix(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    bIsTo,
    IN RvChar*                   pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

bIsTo

Indicates whether a header is a To header (RV_TRUE) or From header (RV_FALSE).

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

RvSipPartyIsEqual()

DESCRIPTION

Compares two Party headers. Party header parameters are considered equal if their URIs match and their header parameters match in name and value. Parameters names and token parameter values are compared ignoring case while quoted-string parameter values are case-sensitive. The tag comparison is performed if both header parameters have a tag value.

SYNTAX

```
RvBool RvSipPartyIsEqual(
    const RvSipPartyHeaderHandle    hHeader,
    const RvSipPartyHeaderHandle    hOtherHeader);
```

PARAMETERS

hHeader

The handle of the Party header.

hOtherHeader

The handle of the Party header with which a comparison is being made.

RETURN VALUES

Returns RV_TRUE if the Party headers being compared are equal. Otherwise returns RV_FALSE.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipPartyHeaderGetStringLength()
- RvSipPartyHeaderGetTag()
- RvSipPartyHeaderSetTag()
- RvSipPartyHeaderGetDisplayName()
- RvSipPartyHeaderSetDisplayName()
- RvSipPartyHeaderGetAddrSpec()
- RvSipPartyHeaderSetAddrSpec()
- RvSipPartyHeaderGetOtherParams()
- RvSipPartyHeaderSetOtherParams()
- RvSipPartyHeaderGetType()
- RvSipPartyHeaderSetType()
- RvSipPartyHeaderGetRpoolString()
- RvSipPartyHeaderSetRpoolString()
- RvSipPartyHeaderGetStrBadSyntax()
- RvSipPartyHeaderSetStrBadSyntax()
- RvSipPartyHeaderSetCompactForm()
- RvSipPartyHeaderGetCompactForm()

RvSipPartyHeaderGetStringLength()

DESCRIPTION

Some of the Party header parameters are kept in a string format, for example, the Party header display name. In order to get such a parameter from the Party header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipPartyHeaderGetStringLength(
    IN RvSipPartyHeaderHandle        hHeader,
    IN RvSipPartyHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Party header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipPartyHeaderGetTag()

DESCRIPTION

Copies the tag parameter of the Party header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipPartyHeaderGetTag(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to include the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetTag()

DESCRIPTION

Sets the tag parameter in the Party header.

SYNTAX

```
RvStatus RvSipPartyHeaderSetTag(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvChar*                   strTag);
```

PARAMETERS

hHeader

The handle of the Party header

strTag

The Tag parameter to be set in the Party header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetDisplayName()

DESCRIPTION

Copies the display name from the Party header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipPartyHeaderGetDisplayName(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetDisplayName()

DESCRIPTION

Sets the display name in the Party header.

SYNTAX

```
RvStatus RvSipPartyHeaderSetDisplayName(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvChar*                   strDisplayName);
```

PARAMETERS

hHeader

The handle of the header.

strDisplayName

The display name to be set in the Party header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetAddrSpec()

DESCRIPTION

The address-spec parameter is held in the Party header as an *address*. This function returns the handle of the *address*.

SYNTAX

```
RvSipAddressHandle RvSipPartyHeaderGetAddrSpec(
    IN RvSipPartyHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Party header.

RETURN VALUES

Returns a handle of the address-spec object, or NULL if the address-spec object does not exist.

RvSipPartyHeaderSetAddrSpec()

DESCRIPTION

Sets the address-spec *address* in the Contact header.

SYNTAX

```
RvStatus RvSipPartyHeaderSetAddrSpec
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvSipAddressHandle        hAddrSpec);
```

PARAMETERS

hHeader

The handle of the Contact header.

hAddrSpec

The handle of the address-spec *address* to be set in the Party header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetOtherParams()

DESCRIPTION

Copies the Party header other-params parameter of the Party header into a given buffer. Not all the Party header parameters have separated parameters in the Party header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipPartyHeaderGetOtherParams(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the Party header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Party header. Not all the Party header parameters have separated parameters in the Party header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipPartyHeaderSetOtherParams(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvChar*                   strOtherParams);
```

PARAMETERS

hHeader

The handle of the Party header

strOtherParams

The other-params string to be set in the Party header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetType()

DESCRIPTION

Returns the type of a Party header, RVSIP_HEADERTYPE_TO or RVSIP_HEADERTYPE_FROM.

SYNTAX

```
RvStatus RvSipPartyHeaderGetType(
    IN  RvSipPartyHeaderHandle    hHeader,
    OUT RvSipHeaderType*          peType);
```

PARAMETERS

hHeader

The handle of the Party header.

peType

The type of the header (To or From).

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetType()

DESCRIPTION

Sets the type of a Party header, RVSIP_HEADERTYPE_TO or RVSIP_HEADERTYPE_FROM.

SYNTAX

```
RvStatus RvSipPartyHeaderSetType(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvSipHeaderType           eType);
```

PARAMETERS

hHEADER

The handle of the Party header.

eTYPE

The type of the header (To or From).

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Party header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipPartyHeaderGetRpoolString(
    IN    RvSipPartyHeaderHandle        hSipPartyHeader,
    IN    RvSipPartyHeaderStringName    eStringName,
    INOUT RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipPartyHeader

The handle of the Party header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Party header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Party header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipPartyHeaderSetRpoolString(
    IN RvSipPartyHeaderHandle        hSipPartyHeader,
    IN RvSipPartyHeaderStringName    eStringName,
    IN RPOOL_Ptr                     *pRpoolPtr);
```

PARAMETERS

hSipPartyHeader

The handle of the Party header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Party header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipPartyHeaderGetStrBadSyntax(
    IN  RvSipPartyHeaderHandle    hHeader,
    IN  RvChar*                   strBuffer,
    IN  RvUint                    bufferLen,
    OUT RvUint*                   actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use this function when you want to send an illegal Party header.

SYNTAX

```
RvStatus RvSipPartyHeaderSetStrBadSyntax(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvChar*                   strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipPartyHeaderSetCompactForm(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    bIsCompact);
```

PARAMETERS

hHeader

The handle of the Party header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipPartyHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the Party header. This flag tells the Stack whether or not to use the compact form Party header during the header encoding.

SYNTAX

```
RvStatus RvSipPartyHeaderGetCompactForm(
    IN RvSipPartyHeaderHandle    hHeader,
    IN RvBool                    *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Party header.

pbIsCompact

Specifies whether or not to use compact form.

RETURN VALUES

Returns RvStatus.

16

RAck Header Functions

RAck header functions enable you to construct, copy, encode, parse, access and change RAck header parameters. This section contains RAck header functions found in the *RvSipRAckHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipRAckHeaderConstructInMsg()
- RvSipRAckHeaderConstruct()
- RvSipRAckHeaderCopy()
- RvSipRAckHeaderEncode()
- RvSipRAckHeaderParse()
- RvSipRAckHeaderParseValue()
- RvSipRAckHeaderFix()
- RvSipRAckHeaderIsResponseNumInitialized()

RvSipRAckHeaderConstructInMsg()

DESCRIPTION

Constructs a RAck header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the header list.

SYNTAX

```
RvStatus RvSipRAckHeaderConstructInMsg(
    IN  RvSipMsgHandle           hSipMsg,
    IN  RvBool                   pushHeaderAtHead,
    OUT RvSipRAckHeaderHandle    *hHeader);
```

PARAMETERS

hSipMsg

The handle of the message.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the RAck header.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone RAck header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipRAckHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipRAckHeaderHandle*     hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed RAck header.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderCopy()

DESCRIPTION

Copies all parameters from a source RAck header to a destination RAck header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipRAckHeaderCopy(
    INOUT RvSipRAckHeaderHandle    hDestination,
    IN    RvSipRAckHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination RAck header.

hSource

The handle of the source RAck header.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderEncode()

DESCRIPTION

Encodes a RAck header to a textual RAck header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal(). The application must free the allocated page, using RPOOL_FreePage(). The allocated page should be freed only if this function returns RV_OK.

SYNTAX

```
RvStatus RvSipRAckHeaderEncode(
    IN  RvSipRAckHeaderHandle    hHeader,
    IN  HRPOOL                   hPool,
    OUT HPAGE*                   hPage,
    OUT RvUint32*                length);
```

PARAMETERS

hHeader

The handle of the RAck header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderParse()

DESCRIPTION

Parses a SIP textual RAck header, such as “RAck:787 1 INVITE”, into a RAck header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipRAckHeaderParse(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

A handle of the RAck header.

buffer

The buffer containing a textual RAck header.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderParseValue()

DESCRIPTION

Parses a SIP textual RAck header value into an RAck header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipRAckHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipRAckHeaderParseValue(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

The handle of the RAck header.

buffer

The buffer containing a textual RAck header value.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderFix()

DESCRIPTION

Fixes a RAck header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipRAckHeaderFix(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvChar*                  pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderIsResponseNumInitialized()

DESCRIPTION

Checks if the response number value in the RAck header is defined.

SYNTAX

```
RvStatus RvSipRAckHeaderIsResponseNumInitialized(
    IN  RvSipRAckHeaderHandle    hHeader,
    OUT RvBool                   *pbInit);
```

PARAMETERS

hHeader

The handle of the RAck header.

pbInit

An indication of the response number definition.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipRAckHeaderGetResponseNum()
- RvSipRAckHeaderSetResponseNum()
- RvSipRAckHeaderGetCSeqHandle()
- RvSipRAckHeaderSetCSeqHandle()
- RvSipRAckHeaderGetStringLength()
- RvSipRAckHeaderGetStrBadSyntax()
- RvSipRAckHeaderSetStrBadSyntax()

RvSipRAckHeaderGetResponseNum()

DESCRIPTION

Gets the response number value from the RAck header.

SYNTAX

```
RvInt32 RvSipRAckHeaderGetResponseNum(
    IN RvSipRAckHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the RAck header.

RETURN VALUES

Returns the response number value.

RvSipRAckHeaderSetResponseNum()

DESCRIPTION

Sets the response number value in the RAck header.

SYNTAX

```
RvStatus RvSipRAckHeaderSetResponseNum(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvInt32                  responseNum);
```

PARAMETERS

hHeader

The handle of the RAck header.

responseNum

Response number value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderGetCSeqHandle()

DESCRIPTION

Gets the Cseq handle from the RAck header.

SYNTAX

```
RvSipCSeqHeaderHandle RvSipRAckHeaderGetCSeqHandle(
    IN RvSipRAckHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the RAck header.

RETURN VALUES

Returns the handle of the CSeq header, or NULL if the CSeq header does not exist.

RvSipRAckHeaderSetCSeqHandle()

DESCRIPTION

Sets a new CSeq header in the RAck header.

SYNTAX

```
RvStatus RvSipRAckHeaderSetCSeqHandle(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvSipCSeqHeaderHandle    hCSeq);
```

PARAMETERS

hHeader

The handle of the RAck header.

hDate

The CSeq handle of be set in the RAck header. If the CSeq handle is NULL, the existing CSeq header is removed from the RAck header.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderGetStringLength()

DESCRIPTION

Some of the RAck header parameters are kept in a string format. In order to get such a parameter from the RAck header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipRAckHeaderGetStringLength(
    IN RvSipRAckHeaderHandle        hHeader,
    IN RvSipRAckHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the RAck header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipRAckHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad RAck header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipRAckHeaderGetStrBadSyntax(
    IN  RvSipRAckHeaderHandle    hHeader,
    IN  RvChar*                  strBuffer,
    IN  RvUint                   bufferLen,
    OUT RvUint*                  actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRAckHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal RAck header.

SYNTAX

```
RvStatus RvSipRAckHeaderSetStrBadSyntax(
    IN RvSipRAckHeaderHandle    hHeader,
    IN RvChar*                  strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipRAckHeaderSetStrBadSyntax()

19

REFER-TO HEADER FUNCTIONS

The Refer-To header functions enable you to construct, copy, encode, parse, access and change Refer-To header parameters. This section contains Refer-To header functions found in the *RvSipReferToHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipReferToHeaderConstructInMsg()
- RvSipReferToHeaderConstruct()
- RvSipReferToHeaderCopy()
- RvSipReferToHeaderEncode()
- RvSipReferToHeaderParse()
- RvSipReferToHeaderParseValue()
- RvSipReferToHeaderFix()

RvSipReferToHeaderConstructInMsg()

DESCRIPTION

Constructs a Refer-To header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipReferToHeaderConstructInMsg(
    IN  RvSipMsgHandle               hSipMsg,
    IN  RvBool                       pushHeaderAtHead,
    OUT RvSipReferToHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Refer-To header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipReferToHeaderConstruct(
    IN  RvSipMsgMgrHandle            hMsgMgr,
    IN  HRPOOL                       hPool,
    IN  HPAGE                        hPage,
    OUT RvSipReferToHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderCopy()

DESCRIPTION

Copies all parameters from a source Refer-To header to a destination Refer-To header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipReferToHeaderCopy(
    INOUT RvSipReferToHeaderHandle    hDestination,
    IN    RvSipReferToHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Refer-To header.

hSource

The handle of the source Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderEncode()

DESCRIPTION

Encodes a Refer-To header to a textual Refer-To header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipReferToHeaderEncode(
    IN  RvSipReferToHeaderHandle    hHeader,
    IN  HRPOOL                      hPool,
    OUT HPAGE*                      hPage,
    OUT RvUint32*                   length);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderParse()

DESCRIPTION

Parses a SIP textual Refer-To header into a Refer-To header. All the textual parameters are placed inside the object. For example, “Refer-To:<sip:charlie@caller.com>”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipReferToHeaderParse(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     buffer);
```

PARAMETERS

hHeader

A handle of the Refer-To header.

buffer

The buffer containing a textual Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderParseValue()

DESCRIPTION

Parses a SIP textual Refer-To header value into an Refer-To header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipReferToHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipReferToHeaderParseValue(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     buffer);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

buffer

The buffer containing a textual Refer-To header value.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderFix()

DESCRIPTION

Fixes a Refer-To header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipReferToHeaderFix(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipReferToHeaderGetStringLength()
- RvSipReferToHeaderGetAddrSpec()
- RvSipReferToHeaderSetAddrSpec()
- RvSipReferToHeaderGetDisplayName()
- RvSipReferToHeaderSetDisplayName()
- RvSipReferToHeaderGetReplacesHeader()
- RvSipReferToHeaderSetReplacesHeader()
- RvSipReferToHeaderGetOtherParams()
- RvSipReferToHeaderSetOtherParams()
- RvSipReferToHeaderGetStrBadSyntax()
- RvSipReferToHeaderSetStrBadSyntax()
- RvSipReferToHeaderGetCompactForm()
- RvSipReferToHeaderSetCompactForm()

RvSipReferToHeaderGetStringLength()

DESCRIPTION

Some of the Refer-To header parameters are kept in a string format, for example, the Refer-To header display name. In order to get such a parameter from the Refer-To header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipReferToHeaderGetStringLength(
    IN RvSipReferToHeaderHandle        hHeader,
    IN RvSipReferToHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipReferToHeaderGetAddrSpec()

DESCRIPTION

Returns the handle of the *address*. The address-spec parameter is held in the Refer-To header as an *address*.

SYNTAX

```
RvSipAddressHandle RvSipReferToHeaderGetAddrSpec(
    IN RvSipReferToHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

RETURN VALUES

Returns a handle of the address-spec object, or NULL if the address-spec object does not exist.

RvSipReferToHeaderSetAddrSpec()

DESCRIPTION

Sets the address-spec *address* in the Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderSetAddrSpec(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvSipAddressHandle          hAddrSpec);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

hAddrSpec

The handle of the address-spec *address* to be set in the Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderGetDisplayName()

DESCRIPTION

Copies the display name from the Refer-To header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipReferToHeaderGetDisplayName(
    IN  RvSipReferToHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderSetDisplayName()

DESCRIPTION

Sets the display name in the Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderSetDisplayName(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     strDisplayName);
```

PARAMETERS

hHeader

The handle of the header.

strDisplayName

The display name to be set in the Refer-To header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderGetReplacesHeader()

DESCRIPTION

Gets the Replaces header from the Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderGetReplacesHeader(
    IN  RvSipReferToHeaderHandle     hHeader,
    OUT RvSipReplacesHeaderHandle    *hReplacesHeader);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

hReplacesHeader

A pointer to the handle of the returned Replaces header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderSetReplacesHeader()

DESCRIPTION

Sets the Replaces header in the Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderSetReplacesHeader(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvSipReplacesHeader         hReplacesHeader);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

hReplacesHeader

The handle of the Replaces header to be set in the Refer-To header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderGetOtherParams()

DESCRIPTION

Copies the other-params parameter of the Refer-To header into a given buffer. The generic parameters of the Refer-To header are referred to as "other-params". They are kept in the object in one concatenated string in the following form: "name=value;name=value...". If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipReferToHeaderGetOtherParams(
    IN  RvSipReferToHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Refer-To header. The generic parameters of the Refer-To header are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipReferToHeaderSetOtherParams(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     strOtherParams);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

strOtherParams

The other-params string to be set in the Refer-To header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Refer-To header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipReferToHeaderGetStrBadSyntax(
    IN  RvSipReferToHeaderHandle    hHeader,
    IN  RvChar*                     strBuffer,
    IN  RvUint                      bufferLen,
    OUT RvUint*                     actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderSetStrBadSyntax(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvChar*                     strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipReferToHeaderSetCompactForm(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvBool                      bIsCompact);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipReferToHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the Refer-To header.

SYNTAX

```
RvStatus RvSipReferToHeaderGetCompactForm(
    IN RvSipReferToHeaderHandle    hHeader,
    IN RvBool                      *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Refer-To header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

20

REFERRED-BY HEADER FUNCTIONS

The Referred-By header functions enable you to construct, copy, encode, parse, access and change Referred-By header parameters. This section contains Referred-By header functions found in the *RvSipReferredByHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipReferredByHeaderConstructInMsg()
- RvSipReferredByheaderConstruct()
- RvSipReferredByHeaderCopy()
- RvSipReferredByHeaderEncode()
- RvSipReferredByHeaderParse()
- RvSipReferredByHeaderParseValue()
- RvSipReferredByHeaderFix()

RvSipReferredByHeaderConstructInMsg()

DESCRIPTION

Constructs a Referred-By header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipReferredByHeaderConstructInMsg(
    IN  RvSipMsgHandle                  hSipMsg,
    IN  RvBool                          pushHeaderAtHead,
    OUT RvSipReferredByHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Referred-By header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByheaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Referred-By header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipReferredByHeaderConstruct(
    IN  RvSipMsgMgrHandle               hMsgMgr,
    IN  HRPOOL                          hPool,
    IN  HPAGE                           hPage,
    OUT RvSipReferredByHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Referred-By header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderCopy()

DESCRIPTION

Copies all parameters from a source Referred-By header to a destination Referred-By header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipReferredByHeaderCopy(
    INOUT RvSipReferredByHeaderHandle    hDestination,
    IN    RvSipReferredByHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Referred-By header.

hSource

The handle of the source Referred-By header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderEncode()

DESCRIPTION

Encodes a Referred-By header to a textual Referred-By header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipReferredByHeaderEncode(
    IN  RvSipReferredByHeaderHandle    hHeader,
    IN  HRPOOL                         hPool,
    OUT HPAGE*                         hPage,
    OUT RvUint32*                      length);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderParse()

DESCRIPTION

Parses a SIP textual Referred-By header into a Referred-By header. All the textual parameters are placed inside the object. For example, “Referred-by: <sip:charlie@caller.com>;ref=<sip:me@com>”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipReferredByHeaderParse(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        buffer);
```

PARAMETERS

hHeader

A handle of the Referred-By header.

buffer

The buffer containing a textual Referred-By header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderParseValue()

DESCRIPTION

Parses a SIP textual Referred-By header value into an Referred-By header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipReferredByHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipReferredByHeaderParseValue(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        buffer);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

buffer

The buffer containing a textual Referred-By header value.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderFix()

DESCRIPTION

Fixes a Referred-By header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipReferredByHeaderFix(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipReferredByHeaderGetStringLength()
- RvSipReferredByHeaderGetDispName()
- RvSipReferredByHeaderSetDispName()
- RvSipReferredByHeaderGetAddrSpec()
- RvSipReferredByHeaderSetAddrSpec()
- RvSipReferredByHeaderGetCidParam()
- RvSipReferredByHeaderSetCidParam()
- RvSipReferredByHeaderGetOtherParams()
- RvSipReferredByHeaderSetOtherParams()
- RvSipReferredByHeaderGetStrBadSyntax()
- RvSipReferredByHeaderSetStrBadSyntax()
- RvSipReferredByHeaderSetCompactForm()
- RvSipReferredByHeaderGetCompactForm()

RvSipReferredByHeaderGetStringLength()

DESCRIPTION

Provides the length of the string to enable allocating an appropriate buffer size before calling the Get function. Some of the Referred-By header parameters are kept in a string format, for example, the referrer display name. In order to get such a parameter from the Referred-By header, your application should supply an adequate buffer to where the string will be copied.

SYNTAX

```
RvUint RvSipReferredByHeaderGetStringLength(
    IN RvSipReferredByHeaderHandle        hHeader,
    IN RvSipReferredByHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipReferredByHeaderGetDispName()

DESCRIPTION

Copies the referrer display name parameter of the Referred-By header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipReferredByHeaderGetDispName(
    IN  RvSipReferredByHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to include the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderSetDispName()

DESCRIPTION

Sets the referred display name parameter in the Referred-By header.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetDispName(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        strDispName);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

strDispName

The display name parameter to be set in the Referred-By header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderGetAddrSpec()

DESCRIPTION

Returns the referrer-URL address-spec.

SYNTAX

```
RvSipAddressHandle RvSipReferredByHeaderGetAddrSpec(
    IN RvSipReferredByHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

RETURN VALUES

Returns a handle of the address-spec object or NULL if this address-spec object does not exist.

RvSipReferredByHeaderSetAddrSpec()

DESCRIPTION

Sets the referrer-URL address-spec in the Referred-By header.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetAddrSpec(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvSipAddressHandle             hAddrSpec,
    IN RvSipReferredByUrl             eUrlEntity);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

hAddrSpec

The handle of the address-spec *address* to be set in the Referred-By header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderGetCidParam()

DESCRIPTION

Copies the referrer cid parameter of the Referred-By header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipReferredByHeaderGetCidParam(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        strBuffer,
    IN RvUint                         bufferLen,
    OUT RvUint*                       actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to include the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderSetCidParam()

DESCRIPTION

Sets the cid parameter in the Referred-By header.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetCidParam(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        strCidParam);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

strCidParam

The cid parameter to be set in the Referred-By header. If NULL is supplied, the existing cid param is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderGetOtherParams()

DESCRIPTION

Copies the Referred-By header other-params parameter of the Referred-By header into a given buffer. Not all the Referred-By header parameters have separated parameters in the Referred-By header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This list also includes the signature parameters. If the value of *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipReferredByHeaderGetOtherParams(
    IN  RvSipReferredByHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter +1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Referred-By header. Not all the Referred-By header parameters have separated parameters in the Referred-By header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This list also includes the signature parameters.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetOtherParams(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        strOtherParams);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

strOtherParams

The other-params string to be set in the Referred-By header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad ReferredBy header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipReferredByHeaderGetStrBadSyntax(
    IN  RvSipReferredByHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Referred-By header.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetStrBadSyntax(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvChar*                        strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipReferredByHeaderSetCompactForm(
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvBool                         bIsCompact);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipReferredByHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the header.

SYNTAX

```
RvStatus RvSipReferredByHeaderGetCompactForm (
    IN RvSipReferredByHeaderHandle    hHeader,
    IN RvBool                         *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Referred-By header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipReferredByHeaderGetCompactForm()

21

RETRY-AFTER HEADER FUNCTIONS

Retry-After header functions enable you to construct, copy, encode, parse, access and change Retry-After header parameters. This section contains Retry-After header functions found in the *RvSipRetryAfterHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipRetryAfterHeaderConstructInMsg()
- RvSipRetryAfterHeaderConstruct()
- RvSipRetryAfterHeaderCopy()
- RvSipRetryAfterHeaderEncode()
- RvSipRetryAfterHeaderParse()
- RvSipRetryAfterHeaderParseValue()
- RvSipRetryAfterHeaderFix()

RvSipRetryAfterHeaderConstructInMsg()

DESCRIPTION

Constructs a Retry-After header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderConstructInMsg(
    IN  RvSipMsgHandle                  hSipMsg,
    IN  RvBool                          pushHeaderAtHead,
    OUT RvSipRetryAfterHeaderHandle*    phHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

phHeader

The handle of the newly constructed Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Retry-After header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderConstruct(
    IN  RvSipMsgMgrHandle               hMsgMgr,
    IN  HRPOOL                          hPool,
    IN  HPAGE                           hPage,
    OUT RvSipRetryAfterHeaderHandle*    phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

phHeader

The handle of the newly constructed Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderCopy()

DESCRIPTION

Copies all parameters from a source Retry-After header to a destination Retry-After header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderCopy(
    INOUT RvSipRetryAfterHeaderHandle    hDestination,
    IN    RvSipRetryAfterHeaderHandle    hSource);
```

PARAMETERS

pDestination

The handle of the destination Retry-After header.

pSource

The handle of the source Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderEncode()

DESCRIPTION

Encodes a Retry-After header to a textual Retry-After header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal(). The application must free the allocated page, using RPOOL_FreePage(). The application must free the allocated page only if this function returns RV_OK.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderEncode(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    IN  HRPOOL                         hPool,
    OUT HPAGE*                         phPage,
    OUT RvUint32*                      pLength);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

hPool

The handle of the specified memory pool.

phPage

The memory page allocated to contain the encoded header.

pLength

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderParse()

DESCRIPTION

Parses a SIP textual Expires header into a Retry-After header. All the textual parameters are placed inside the object. For example, “RetryAfter: Thu, 01 Dec 2040 16:00:00 GMT”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderParse(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        strBuffer);
```

PARAMETERS

hHeader

A handle of the Retry-After header.

strBuffer

The buffer containing a textual Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderParseValue()

DESCRIPTION

Parses a SIP textual Retry-After header value into an Retry-After header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipRetryAfterHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderParseValue(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        buffer);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

buffer

The buffer containing a textual Retry-After header value.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderFix()

Fixes a Retry-After header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderFix(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipRetryAfterHeaderGetStringLength()
- RvSipRetryAfterHeaderGetFormat()
- RvSipRetryAfterHeaderGetDeltaSeconds()
- RvSipRetryAfterHeaderSetDeltaSeconds()
- RvSipRetryAfterHeaderGetDateHandle()
- RvSipRetryAfterHeaderSetDate()
- RvSipRetryAfterHeaderGetDuration()
- RvSipRetryAfterHeaderSetDuration()
- RvSipRetryAfterHeaderGetStrComment()
- RvSipRetryAfterHeaderSetStrComment()
- RvSipRetryAfterHeaderGetRetryAfterParams()
- RvSipRetryAfterHeaderSetRetryAfterParams()
- RvSipRetryAfterHeaderGetStrBadSyntax()
- RvSipRetryAfterHeaderSetStrBadSyntax()

RvSipRetryAfterHeaderGetStringLength()

DESCRIPTION

Some of the Retry-After header parameters are kept in string format, such as the comment. In order to get such a parameter from the Retry-After header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipRetryAfterHeaderGetStringLength(
    IN RvSipRetryAfterHeaderHandle        hHeader,
    IN RvSipRetryAfterHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipRetryAfterHeaderGetFormat()

DESCRIPTION

Gets the format of the Retry-After header: Undefined, Date or Delta-seconds.

SYNTAX

```
RvSipExpiresFormat RvSipRetryAfterHeaderGetFormat(
    IN RvSipRetryAfterHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

RETURN VALUES

Returns the format enumeration.

RvSipRetryAfterHeaderGetDeltaSeconds()

DESCRIPTION

Gets the delta-seconds integer of the Retry-After header. If the delta-seconds integer is not set, UNDEFINED is returned.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderGetDeltaSeconds(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    OUT RvUint32                       *pDeltaSeconds);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

pDeltaSeconds

The delta-seconds integer.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderSetDeltaSeconds()

DESCRIPTION

Sets the delta seconds integer of the Retry-After header. Changes the Retry-After format to delta-seconds. If the given delta-seconds is UNDEFINED, the delta-seconds of the Retry-After header is removed and the format is changed to UNDEFINED.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetDeltaSeconds(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvUint32                       deltaSeconds);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

deltaSeconds

The delta seconds to be set to the Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderGetDateHandle()

DESCRIPTION

Gets the date handle of the Retry-After header.

SYNTAX

```
RvSipDateHeaderHandle RvSipRetryAfterHeaderGetDateHandle(
    IN RvSipRetryAfterHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

RETURN VALUES

Returns the handle of the Date header, or NULL if the Date header does not exist.

RvSipRetryAfterHeaderSetDate()

DESCRIPTION

Sets a new Date header in the Retry-After header and changes the Retry-After format to date. (The function allocates a Date header, and copies the given *hDate* object to it).

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetDate(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvSipDateHeaderHandle          hDate);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

hDate

The date handle of be set to the Retry-After header. If the date handle is NULL, the existing Date header is removed from the Expires header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderGetDuration()

DESCRIPTION

Gets the duration integer of the Retry-After header. If the duration integer is not set, UNDEFINED is returned.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderGetDuration(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    OUT RvInt32                        *pDuration);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

pDuration

The duration integer.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderSetDuration()

DESCRIPTION

Sets the duration integer of the Retry-After header. If the given duration is UNDEFINED, the duration of the Retry-After header is removed.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetDuration(
    IN RvSipRetryAfterHeaderHandlev    hHeader,
    IN RvInt32                         duration);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

duration

The duration to be set to the Retry-After header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderGetStrComment()

DESCRIPTION

Copies the str-comment parameter of the Retry-After header into a given buffer. If the value of *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderGetStrComment(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter +1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderSetStrComment()

DESCRIPTION

Sets the str-comment parameter in the Retry-After header.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetStrComment(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        strComment);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

strComment

The strComment string to be set in the Retry-After header. If NULL is supplied, the existing strComment parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderGetRetryAfterParams()

DESCRIPTION

Copies the retry-after-params parameter of the Retry-After header into a given buffer. Not all the Retry-After header parameters have separated parameters in the Retry-After header. Parameters with no specific parameters are referred to as RetryAfterParams. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

If the value of *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderGetRetryAfterParams(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter +1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderSetRetryAfterParams()

DESCRIPTION

Sets the retry-after-params parameter in the Retry-After header. Not all the Retry-After header parameters have separated parameters in the Retry-After header. Parameters with no specific parameters are referred to as “retry-after-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetRetryAfterParams(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        strRetryAfterParams);
```

PARAMETERS

hHeader

The handle of the Retry-After header.

strRetryAfterParams

The retry-after-params string to be set in the Retry-After header. If NULL is supplied, the existing retry-after-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Retry-After header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderGetStrBadSyntax(
    IN  RvSipRetryAfterHeaderHandle    hHeader,
    IN  RvChar*                        strBuffer,
    IN  RvUint                         bufferLen,
    OUT RvUint*                        actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRetryAfterHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Retry-After header.

SYNTAX

```
RvStatus RvSipRetryAfterHeaderSetStrBadSyntax(
    IN RvSipRetryAfterHeaderHandle    hHeader,
    IN RvChar*                        strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

22

ROUTE-HOP HEADER FUNCTIONS

The Route-Hop header represents both Route and Record-Route SIP headers. The Route-Hop header functions enable you to construct, copy, encode, parse, access and change Route-Hop header parameters. This section contains Route-Hop header functions found in the *RvSipRouteHopHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipRouteHopHeaderConstructInMsg()
- RvSipRouteHopHeaderConstruct()
- RvSipRouteHopHeaderCopy()
- RvSipRouteHopHeaderEncode()
- RvSipRouteHopHeaderParse()
- RvSipRouteHopHeaderParseValue()
- RvSipRouteHopHeaderFix()

RvSipRouteHopHeaderConstructInMsg()

DESCRIPTION

Constructs a Route-Hop header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list. Use RvSipRouteHopHeaderSetHeaderType() to determine whether the object represents a Route, Record-Route, Path or Service-Route header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderConstructInMsg(
    IN  RvSipMsgHandle                hSipMsg,
    IN  RvBool                        pushHeaderAtHead,
    OUT RvSipRouteHopHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Route header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Route-Hop header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned. Use RvSipRouteHopHeaderSetHeaderType() to determine whether the object represents a Route, Record-Route, Path or Service-Route header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderConstruct(
    IN  RvSipMsgMgrHandle             hMsgMgr,
    IN  HRPOOL                        hPool,
    IN  HPAGE                         hPage,
    OUT RvSipRouteHopHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Route-Hop header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderCopy()

DESCRIPTION

Copies all parameters from a source Route-Hop header to a destination Route-Hop header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipRouteHopHeaderCopy(
    INOUT RvSipRouteHopHeaderHandle    hDestination,
    IN    RvSipRouteHopHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Route-Hop header.

hSource

The handle of the source Route-Hop header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderEncode()

DESCRIPTION

Encodes a Route-Hop header to a textual Route or Record-Route header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipRouteHopHeaderEncode(
    IN  RvSipRouteHopHeaderHandle    hHeader,
    IN  HRPOOL                       hPool,
    OUT HPAGE*                       hPage,
    OUT RvUint32*                    length);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderParse()

DESCRIPTION

Parses a SIP textual Route, Record-Route, Path or Service-Route header. All the textual parameters are placed inside the object. For example,

“Record-Route: <sip:UserB@there.com;maddr=ss1.wcom.com>”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipRouteHopHeaderParse(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      buffer);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

buffer

The buffer containing a textual Route-Hop header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderParseValue()

DESCRIPTION

Parses a SIP textual Route-Hop header value into a Route-Hop header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipRouteHopHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipRouteHopHeaderParseValue(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      buffer);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

buffer

The buffer containing a textual Route-Hop header value.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderFix()

DESCRIPTION

Fixes a Route-Hop header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipRouteHopHeaderFix(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipRouteHopHeaderGetStringLength()
- RvSipRouteHopHeaderGetDisplayName()
- RvSipRouteHopHeaderSetDisplayName()
- RvSipRouteHopHeaderGetAddrSpec()
- RvSipRouteHopHeaderSetAddrSpec()
- RvSipRouteHopHeaderGetOtherParams()
- RvSipRouteHopHeaderSetOtherParams()
- RvSipRouteHopHeaderGetHeaderType()
- RvSipRouteHopHeaderSetHeaderType()
- RvSipRouteHopHeaderGetRpoolString()
- RvSipRouteHopHeaderSetRpoolString()
- RvSipRouteHopHeaderGetStrBadSyntax()
- RvSipRouteHopHeaderSetStrBadSyntax()

RvSipRouteHopHeaderGetStringLength()

DESCRIPTION

Provides the length of string parameters to enable allocating an appropriate buffer size before calling the Get function. Some of the Route-Hop header parameters are kept in a string format, for example, the Route header display name. In order to get such a parameter from the Route header, your application should supply an adequate buffer to where the string will be copied.

SYNTAX

```
RvUint RvSipRouteHopHeaderGetStringLength(
    IN RvSipRouteHopHeaderHandle        hHeader,
    IN RvSipRouteHopHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipRouteHopHeaderGetDisplayName()

DESCRIPTION

Copies the display name from the Route-Hop header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipRouteHopHeaderGetDisplayName(
    IN  RvSipRouteHopHeaderHandle    hHeader,
    IN  RvChar*                      strBuffer,
    IN  RvUint                       bufferLen,
    OUT RvUint*                      actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderSetDisplayName()

DESCRIPTION

Sets the display name in the Route-Hop header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetDisplayName(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      strDisplayName);
```

PARAMETERS

hHeader

The handle of the header.

strDisplayName

The display name to be set in the Route-Hop header. If NULL is supplied, the existing display name is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderGetAddrSpec()

DESCRIPTION

The address-spec parameter is held in the Route-Hop header as an *address*. This function returns the handle of the *address*.

SYNTAX

```
RvSipAddressHandle RvSipRouteHopHeaderGetAddrSpec(
    IN RvSipRouteHopHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

RETURN VALUES

Returns a handle of the address-spec object, or NULL if the address-spec object does not exist.

RvSipRouteHopHeaderSetAddrSpec()

DESCRIPTION

Sets the address-spec *address* in the Route-Hop header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetAddrSpec(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvSipAddressHandle           hAddrSpec);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

hAddrSpec

The handle of the address-spec *address* to be set in the Route-Hop header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderGetOtherParams()

DESCRIPTION

Not all the Route-Hop header parameters have separated parameters in the Route-Hop header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function copies the Route-Hop header other-params parameter of the Route-Hop header into a given buffer. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipRouteHopHeaderGetOtherParams(
    IN  RvSipRouteHopHeaderHandle    hHeader,
    IN  RvChar*                      strBuffer,
    IN  RvUint                       bufferLen,
    OUT RvUint*                      actualLen);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderSetOtherParams()

DESCRIPTION

Not all the Route-Hop header parameters have separated parameters in the Route-Hop header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. This function sets the other-params parameter in the Route-Hop header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetOtherParams(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      strOtherParams);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

strOtherParams

The other-params string to be set in the Route-Hop header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderGetHeaderType()

DESCRIPTION

Gets the header type enumeration from the Route-Hop header. The Route-Hop header can represent a Route or Record-Route header.

SYNTAX

```
RvSipRouteHopHeaderType RvSipRouteHopHeaderGetHeaderType(
    IN RvSipRouteHopHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

RETURN VALUES

Returns the Route-Hop header type enumeration from the Route-Hop header.

RvSipRouteHopHeaderSetHeaderType()

DESCRIPTION

Sets the header type in the Route-Hop header. The Route-Hop header can represent a Route or Record-Route header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetHeaderType(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvSipRouteHopHeaderType      eHeaderType);
```

PARAMETERS

hHeader

The handle of the Route-Hop header.

eHeaderType

The header type to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderGetRpoolString()

DESCRIPTION

Copy a string parameter from the Route-Hop header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipRouteHopHeaderGetRpoolString(
   IN    RvSipRouteHopHeaderHandle       hSipRouteHopHeader,
   IN    RvSipRouteHopHeaderStringName   eStringName,
   INOUT RPOOL_Ptr                       *pRpoolPtr);
```

PARAMETERS

hSipRouteHopHeader

The handle of the Route-Hop header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Route-Hop header.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Route-Hop header. The given string is located in an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetRpoolString(
    IN RvSipRouteHopHeaderHandle        hSipRouteHopHeader,
    IN RvSipRouteHopHeaderStringName    eStringName,
    IN RPOOL_Ptr                        *pRpoolPtr);
```

PARAMETERS

hSipRouteHopHeader

The handle of the Route-Hop header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Route-Hop header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipRouteHopHeaderGetStrBadSyntax(
    IN  RvSipRouteHopHeaderHandle    hHeader,
    IN  RvChar*                      strBuffer,
    IN  RvUint                       bufferLen,
    OUT RvUint*                      actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRouteHopHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Route-Hop header.

SYNTAX

```
RvStatus RvSipRouteHopHeaderSetStrBadSyntax(
    IN RvSipRouteHopHeaderHandle    hHeader,
    IN RvChar*                      strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipRouteHopHeaderSetStrBadSyntax()

23

RSEQ HEADER FUNCTIONS

RSeq header functions enable you to construct, copy, encode, parse, access and change RSeq header parameters. This section contains RSeq header functions found in the *RvSipRSeqHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipRSeqHeaderConstructInMsg()
- RvSipRSeqHeaderConstruct()
- RvSipRSeqHeaderCopy()
- RvSipRSeqHeaderEncode()
- RvSipRSeqHeaderParse()
- RvSipRSeqHeaderParseValue()
- RvSipRSeqHeaderFix()

RvSipRSeqHeaderConstructInMsg()

DESCRIPTION

Constructs an RSeq header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the header list.

SYNTAX

```
RvStatus RvSipRSeqHeaderConstructInMsg(
    IN  RvSipMsgHandle           hSipMsg,
    IN  RvBool                   pushHeaderAtHead,
    OUT RvSipRSeqHeaderHandle    *hHeader);
```

PARAMETERS

hSipMsg

The handle of the message related to the new header.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed RSeq header.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone RSeq header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipRSeqHeaderConstruct(
    IN  RvSipMsgMgrHandle          hMsgMgr,
    IN  HRPOOL                     hPool,
    IN  HPAGE                      hPage,
    OUT RvSipRSeqHeaderHandle*     hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed RSeq header.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderCopy()

DESCRIPTION

Copies all parameters from a source RSeq header to a destination RSeq header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipRSeqHeaderCopy(
    INOUT RvSipRSeqHeaderHandle    hDestination,
    IN    RvSipRSeqHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination RSeq header.

hSource

The handle of the source RSeq header.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderEncode()

DESCRIPTION

Encodes an RSeq header to a textual RSeq header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal(). The application must free the allocated page using RPOOL_FreePage(). The allocated page should be freed only if this function returns RV_OK.

SYNTAX

```
RvStatus RvSipRSeqHeaderEncode(
    IN  RvSipRSeqHeaderHandle    hHeader,
    IN  HRPOOL                   hPool,
    OUT HPAGE*                   hPage,
    OUT RvUint32*                length);
```

PARAMETERS

hHeader

The handle of the RSeq header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderParse()

DESCRIPTION

Parses a SIP textual RSeq header, such as “RSeq:43”, into an RSeq header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipRSeqHeaderParse(
    IN RvSipRSeqHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

The handle of the RSeq header.

buffer

The buffer containing a textual RSeq header.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderParseValue()

DESCRIPTION

Parses a SIP textual RSeq header value into an RSeq header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipRSeqHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipRSeqHeaderParseValue(
    IN RvSipRSeqHeaderHandle    hHeader,
    IN RvChar*                  buffer);
```

PARAMETERS

hHeader

The handle of the RSeq header.

buffer

The buffer containing a textual RSeq header value.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderFix()

DESCRIPTION

Fixes a RSeq header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string.You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipRSeqHeaderFix(
    IN RvSipRSeqHeaderHandle    hHeader,
    IN RvChar*                  pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipRSeqHeaderGetResponseNum()
- RvSipRSeqHeaderSetResponseNum()
- RvSipRSeqHeaderGetStringLength()
- RvSipRSeqHeaderGetStrBadSyntax()
- RvSipRSeqHeaderSetStrBadSyntax()
- RvSipRSeqHeaderIsResponseNumInitialized()

RvSipRSeqHeaderGetResponseNum()

DESCRIPTION

Gets the response number value from the RSeq header.

SYNTAX

```
RvInt32 RvSipRSeqHeaderGetResponseNum(
    IN RvSipRSeqHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the RSeq header.

RETURN VALUES

Returns the response number value, or UNDEFINED if the number is not set.

RvSipRSeqHeaderSetResponseNum()

DESCRIPTION

Sets the response num value in the RSeq header.

SYNTAX

```
RvStatus RvSipRSeqHeaderSetResponseNum(
    IN RvSipRSeqHeaderHandle    hHeader,
    IN RvInt32                  responseNum);
```

PARAMETERS

hHeader

The handle of the RSeq header.

responseNum

Response number value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderGetStringLength()

DESCRIPTION

Some of the RSeq header parameters are kept in a string format. In order to get such a parameter from the RSeq header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipRSeqHeaderGetStringLength(
    IN RvSipRSeqHeaderHandle        hHeader,
    IN RvSipRSeqHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the RSeq header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipRSeqHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad RSeq header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipRSeqHeaderGetStrBadSyntax(
    IN  RvSipRSeqHeaderHandle    hHeader,
    IN  RvChar*                  strBuffer,
    IN  RvUint                   bufferLen,
    OUT RvUint*                  actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal RSeq header.

SYNTAX

```
RvStatus RvSipRSeqHeaderSetStrBadSyntax(
    IN RvSipRSeqHeaderHandle    hHeader,
    IN RvChar*                  strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipRSeqHeaderIsResponseNumInitialized()

DESCRIPTION

Gets the flag that indicates if the response number was initialized.

SYNTAX

```
RvStatus RvSipRSeqHeaderIsResponseNumInitialized(
    IN  RvSipRSeqHeaderHandle    hHeader,
    OUT RvBool                   *pbInit);
```

PARAMETERS

hHeader

The handle of the Allow-Events header.

pbInit

Specifies whether or not the response number was initialized.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipRSeqHeaderIsResponseNumInitialized()

24

SESSION-EXPIRES HEADER FUNCTIONS

The Session-Expires header functions enable you to construct, copy, encode, parse, access and change Session-Expires header parameters. This section contains Session-Expires header functions found in the *RvSipSessionExpiresHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipSessionExpiresHeaderConstructInMsg()
- RvSipSessionExpiresHeaderConstruct()
- RvSipSessionExpiresHeaderCopy()
- RvSipSessionExpiresHeaderEncode()
- RvSipSessionExpiresHeaderParse()
- RvSipSessionExpiresHeaderParseValue()
- RvSipSessionExpiresHeaderFix()

RvSipSessionExpiresHeaderConstructInMsg()

DESCRIPTION

Constructs a Session-Expires header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderConstructInMsg(
   IN  RvSipMsgHandle                    hSipMsg,
   IN  RvBool                            pushHeaderAtHead,
   OUT RvSipSessionExpiresHeaderHandle*  phHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

phHeader

The handle of the newly constructed Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Session-Expires header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderConstruct(
    IN  RvSipMsgMgrHandle                   hMsgMgr,
    IN  HRPOOL                              hPool,
    IN  HPAGE                               hPage,
    OUT RvSipSessionExpiresHeaderHandle*    phHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

phHeader

The handle of the newly constructed Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderCopy()

DESCRIPTION

Copies all parameters from a source Session-Expires header to a destination Session-Expires header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderCopy(
    INOUT RvSipSessionExpiresHeaderHandle    hDestination,
    IN    RvSipSessionExpiresHeaderHandle    hSource);
```

PARAMETERS

pDestination

The handle of the destination Session-Expires header.

pSource

The handle of the source Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderEncode()

DESCRIPTION

Encodes a Session-Expires header to a textual Session-Expires header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal(). The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderEncode(
    IN  RvSipSessionExpiresHeaderHandle    hHeader,
    IN  HRPOOL                             hPool,
    OUT HPAGE*                             phPage,
    OUT RvUint32*                          pLength);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

hPool

The handle of the specified memory pool.

phPage

The memory page allocated to contain the encoded header.

pLength

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderParse()

DESCRIPTION

Parses a SIP textual Session-Expires header into a Session-Expires header. All the textual parameters are placed inside the object. For example, “Session Expires: 3600;refresher=uac”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderParse(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvChar*                            strBuffer);
```

PARAMETERS

hHeader

A handle of the Session-Expires header.

strBuffer

The buffer containing a textual Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderParseValue()

DESCRIPTION

Parses a SIP textual Session-Expires header value into a Session-Expires header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipSessionExpiresHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderParseValue(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvChar*                            buffer);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

buffer

The buffer containing a textual Session-Expires header value.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderFix()

DESCRIPTION

Fixes a Session-Expires header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderFix(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvChar*                            pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipSessionExpiresHeaderGetStringLength()
- RvSipSessionExpiresHeaderGetDeltaSeconds()
- RvSipSessionExpiresHeaderSetDeltaSeconds()
- RvSipSessionExpiresHeaderGetRefresherType()
- RvSipSessionExpiresHeaderSetRefresherType()
- RvSipSessionExpiresHeaderGetOtherParams()
- RvSipSessionExpiresHeaderSetOtherParams()
- RvSipSessionExpiresHeaderGetStrBadSyntax()
- RvSipSessionExpiresHeaderSetStrBadSyntax()
- RvSipSessionExpiresHeaderSetCompactForm()
- RvSipSessionExpiresHeaderGetCompactForm()

RvSipSessionExpiresHeaderGetStringLength()

DESCRIPTION

The other-params of Session-Expires header parameters are kept in a string format. In order to get such a parameter from the Session-Expires header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipSessionExpiresHeaderGetStringLength(
    IN RvSipSessionExpiresHeaderHandle        hHeader,
    IN RvSipSessionExpiresHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipSessionExpiresHeaderGetDeltaSeconds()

DESCRIPTION

Gets the delta-seconds integer of the Session-Expires header. If the delta-seconds integer is not set, UNDEFINED is returned.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderGetDeltaSeconds(
    IN  RvSipSessionExpiresHeaderHandle    hHeader,
    OUT RvInt32                            *pDeltaSeconds);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

pDeltaSeconds

The delta-seconds integer.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderSetDeltaSeconds()

DESCRIPTION

Sets the delta seconds integer of the Session-Expires header. If the given delta-seconds is UNDEFINED, the delta-seconds of the Session-Expires header is removed.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderSetDeltaSeconds(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvInt32                            deltaSeconds);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

deltaSeconds

The delta-seconds to be set to the Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderGetRefresherType()

DESCRIPTION

Gets the refresher type of the Session-Expires header. If the refresher type is not set, RVSIP_SESSION_EXPIRES_REFRESHER_NONE is returned.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderGetRefresherType(
  IN  RvSipSessionExpiresHeaderHandle     hHeader,
  OUT RvSipSessionExpiresRefresherType    *peRefresherType);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

peRefresherType

The refresher type.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderSetRefresherType()

DESCRIPTION

Sets the refresher type of the Session-Expires header.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderSetRefresherType(
    IN RvSipSessionExpiresHeaderHandle     hHeader,
    IN RvSipSessionExpiresRefresherType    eRefresherType);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

deltaSeconds

The refresher type to be set to the Session-Expires header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderGetOtherParams()

DESCRIPTION

Copies the other-params string from the Session-Expires header into a given buffer. Not all the Session-Expires header parameters have separated parameters in the Session-Expires header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderGetOtherParams(
    IN  RvSipSessionExpiresHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderSetOtherParams()

DESCRIPTION

Sets the other-params string in the Session-Expires header. The given string is copied to the Session-Expires header. Not all the Session-Expires header parameters have separated parameters in the Session-Expires header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderSetOtherParams(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvChar                             *otherParams);
```

PARAMETERS

hHeader

The handle of the header.

otherParams

The other-params string to be set in the Session-Expires header. If NULL is supplied, the existing parameters string is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Session-Expires header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderGetStrBadSyntax(
    IN  RvSipSessionExpiresHeaderHandle    hHeader,
    IN  RvChar*                            strBuffer,
    IN  RvUint                             bufferLen,
    OUT RvUint*                            actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Session-Expires header.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderSetStrBadSyntax(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvChar*                            strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderSetCompactForm(
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvBool                             bIsCompact);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipSessionExpiresHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the header.

SYNTAX

```
RvStatus RvSipSessionExpiresHeaderGetCompactForm (
    IN RvSipSessionExpiresHeaderHandle    hHeader,
    IN RvBool                             pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Session-Expires header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

25

SUBSCRIPTION-STATE HEADER FUNCTIONS

The Subscription-State header functions enable you to construct, copy, encode, parse, access and change Subscription-State header parameters. This section contains Subscription-State header API functions found in the *RvSipSubscriptionStateHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipSubscriptionStateHeaderConstructInMsg()
- RvSipSubscriptionStateHeaderConstruct()
- RvSipSubscriptionStateHeaderCopy()
- RvSipSubscriptionStateHeaderEncode()
- RvSipSubscriptionStateHeaderParse()
- RvSipSubscriptionStateHeaderParseValue()
- RvSipSubscriptionStateHeaderFix()

RvSipSubscriptionStateHeaderConstructInMsg()

DESCRIPTION

Constructs a Subscription-State header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderConstructInMsg(
  IN  RvSipMsgHandle                      hSipMsg,
  IN  RvBool                              pushHeaderAtHead,
  OUT RvSipSubscriptionStateHeaderHandle* hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Subscription-State header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderConstruct(
    IN  RvSipMsgMgrHandle                    hMsgMgr,
    IN  HRPOOL                               hPool,
    IN  HPAGE                                hPage,
    OUT RvSipSubscriptionStateHeaderHandle*  hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderCopy()

DESCRIPTION

Copies all parameters from a source Subscription-State header to a destination Subscription-State header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderCopy(
  INOUT RvSipSubscriptionStateHeaderHandle    hDestination,
  IN    RvSipSubscriptionStateHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Subscription-State header.

hSource

The handle of the source Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderEncode()

DESCRIPTION

Encodes a Subscription-State header to a textual Subscription-State header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderEncode(
    IN  RvSipSubscriptionStateHeaderHandle    hHeader,
    IN  HRPOOL                                hPool,
    OUT HPAGE*                                hPage,
    OUT RvUint32*                             length);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderParse()

DESCRIPTION

Parses a SIP textual Subscription-State header into a Subscription-State header. An example of a SIP textual Subscription-State header is “Subscription-State: “active”;expires=100”. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderParse(
    IN RvSipSubscriptionStateHeaderHandle    hHeader,
    IN RvChar*                               buffer);
```

PARAMETERS

hHeader

A handle of the Subscription-State header.

buffer

A buffer containing a textual Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderParseValue()

DESCRIPTION

Parses a SIP textual Subscription-State header value into a Subscription-State header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipSubscriptionStateHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderParseValue(
    IN RvSipSubscriptionStateHeaderHandle    hHeader,
    IN RvChar*                               buffer);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

buffer

The buffer containing a textual Subscription-State header value.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderFix()

DESCRIPTION

Fixes a Subscription-State header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderFix(
    IN RvSipSubscriptionStateHeaderHandle     hHeader,
    IN RvChar*                                pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipSubscriptionStateHeaderGetStringLength()
- RvSipSubscriptionStateHeaderGetSubstate()
- RvSipSubscriptionStateHeaderGetStrSubstate()
- RvSipSubscriptionStateHeaderSetSubstate()
- RvSipSubscriptionStateHeaderGetReason()
- RvSipSubscriptionStateHeaderGetStrReason()
- RvSipSubscriptionStateHeaderSetReason()
- RvSipSubscriptionStateHeaderGetOtherParams()
- RvSipSubscriptionStateHeaderSetOtherParams()
- RvSipSubscriptionStateHeaderGetExpiresParam()
- RvSipSubscriptionStateHeaderSetExpiresParam()
- RvSipSubscriptionStateHeaderGetRetryAfter()
- RvSipSubscriptionStateHeaderSetRetryAfter()
- RvSipSubscriptionStateHeaderGetStrBadSyntax()
- RvSipSubscriptionStateHeaderSetStrBadSyntax()

RvSipSubscriptionStateHeaderGetStringLength()

DESCRIPTION

Some of the Subscription-State header parameters are kept in a string format, for example, the Subscription-State header “other-params”. In order to get such a parameter from the Subscription-State header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipSubscriptionStateHeaderGetStringLength(
   IN RvSipSubscriptionStateHeaderHandle        hHeader,
   IN RvSipSubscriptionStateHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipSubscriptionStateHeaderGetSubstate()

DESCRIPTION

Gets the substate enumeration value of the subscription-state object. If RVSIP_SUBSCRIPTION_SUBSTATE_OTHER is returned, you can receive the substate string using RvSipSubscriptionStateHeaderGetSubstate().

SYNTAX

```
RvSipSubscriptionSubstate
                    RvSipSubscriptionStateHeaderGetSubstate(
    IN RvSipSubscriptionStateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

RETURN VALUES

Returns an enumeration value of substate.

RvSipSubscriptionStateHeaderGetStrSubstate()

DESCRIPTION

Copies the substate string from the Subscription-State header into a given buffer. Use this function when the substate enumeration returned by RvSipSubscriptionStateHeaderGetSubstate() is RVSIP_SUBSCRIPTION_SUBSTATE_OTHER. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderGetStrSubstate(
    IN  RvSipSubscriptionStateHeaderHandle    hHeader,
    IN  RvChar*                               strBuffer,
    IN  RvUint                                bufferLen,
    OUT RvUint*                               actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderSetSubstate()

DESCRIPTION

Sets the substate parameter in the Subscription-State header. If the enumeration given by *eSubstate* is RVSIP_SUBSCRIPTION_SUBSTATE_OTHER, sets the substate string given by strSubstate in the Subscription-State header. Use RVSIP_SUBSCRIPTION_SUBSTATE_OTHER when the substate you wish to set to the Subscription-State header does not have a matching enumeration value in the RvSipSubscriptionSubstate enumeration.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetSubstate(
    IN RvSipSubscriptionStateHeaderHandle    hHeader,
    IN RvSipSubscriptionSubstate             eSubstate,
    IN RvChar                                *strSubstate);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

eSubstate

The substate enumeration to be set in the Subscription-State header. If RVSIP_SUBSCRIPTION_SUBSTATE_UNDEFINED is supplied, the existing substate is removed from the header.

strSubstate

The substate string to be set in the Subscription-State header.

(Relevant when *eSubstate* is RVSIP_SUBSCRIPTION_SUBSTATE_OTHER.)

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderGetReason()

DESCRIPTION

Gets the reason enumeration value of the subscription-state object. If RVSIP_SUBSCRIPTION_REASON_OTHER is returned, you can receive the reason string using RvSipSubscriptionStateHeaderGetStrReason().

SYNTAX

```
RvSipSubscriptionReason
  RvSipSubscriptionStateHeaderGetReason(
    IN RvSipSubscriptionStateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

RETURN VALUES

Returns an enumeration value of the substate.

RvSipSubscriptionStateHeaderGetStrReason()

DESCRIPTION

Copies the reason string from the Subscription-State header into a given buffer. Use this function when the reason enumeration returned by RvSipSubscriptionStateHeaderGetReason() is RVSIP_SUBSCRIPTION_REASON_OTHER. If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderGetStrReason(
    IN  RvSipSubscriptionStateHeaderHandle    hHeader,
    IN  RvChar*                               strBuffer,
    IN  RvUint                                bufferLen,
    OUT RvUint*                               actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1 to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderSetReason()

DESCRIPTION

Sets the reason parameter in the Subscription-State header. If the enumeration given by *eReason* is RVSIP_SUBSCRIPTION_REASON_OTHER, sets the reason string given by *strReason* in the Subscription-State header. Use RVSIP_SUBSCRIPTION_REASON_OTHER when the substate you wish to set to the Subscription-State header does not have a matching enumeration value in the RvSipSubscriptionReason enumeration.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetReason(
    IN RvSipSubscriptionStateHeaderHandle    hHeader,
    IN RvSipSubscriptionReason               eReason,
    IN RvChar                                *strReason);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

eReason

The reason enumeration to be set in the Subscription-State header. If RVSIP_SUBSCRIPTION_REASON_UNDEFINED is supplied, the existing reason is removed from the header.

strReason

The reason string to be set in the Subscription-State header.

(Relevant when eReason is RVSIP_SUBSCRIPTION_REASON_OTHER.)

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderGetOtherParams()

DESCRIPTION

Copies the Subscription-State header other-params parameter of the Subscription-State header into a given buffer. Not all the Subscription-State header parameters have separated parameters in the Subscription-State header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

If *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderGetOtherParams(
    IN  RvSipSubscriptionStateHeaderHandle    hHeader,
    IN  RvChar*                               strBuffer,
    IN  RvUint                                bufferLen,
    OUT RvUint*                               actualLen);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Subscription-State header. Not all the Subscription-State header parameters have separated parameters in the Subscription-State header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetOtherParams(
   IN RvSipSubscriptionStateHeaderHandle   hHeader,
   IN RvChar*                              strOtherParam);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

strOtherParam

The extended parameters parameter to be set in the Subscription-State header. If NULL is supplied, the existing extended parameters parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderGetExpiresParam()

DESCRIPTION

The Subscription-State header may contain an expires parameter. This function returns the expires parameter value.

SYNTAX

```
RvInt32 RvSipSubscriptionStateHeaderGetExpiresParam(
    IN RvSipSubscriptionStateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

RETURN VALUES

Returns the expires parameter value, UNDEFINED if not exists.

RvSipSubscriptionStateHeaderSetExpiresParam()

DESCRIPTION

Sets an expires parameter in the Subscription-State header.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetExpiresParam(
    IN RvSipSubscriptionStateHeaderHandle    hHeader,
    IN RvInt32                               expiresVal);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

hExpires

The expires value to be set in the Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderGetRetryAfter()

DESCRIPTION

The Subscription-State header may contain a retry-after parameter. This function returns this parameter value.

SYNTAX

```
RvInt32 RvSipSubscriptionStateHeaderGetRetryAfter(
    IN RvSipSubscriptionStateHeaderHandle    hHeader);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

RETURN VALUES

Returns the retry-after parameter value, UNDEFINED if not exists.

RvSipSubscriptionStateHeaderSetRetryAfter()

DESCRIPTION

Sets a retry-after parameter in the Subscription-State header.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetRetryAfter(
   IN RvSipSubscriptionStateHeaderHandle    hHeader,
   IN RvInt32                               retryAfterVal);
```

PARAMETERS

hHeader

The handle of the Subscription-State header.

hExpires

The retry-after value to be set in the Subscription-State header.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Subscription-State header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderGetStrBadSyntax(
    IN  RvSipSubscriptionStateHeaderHandle    hHeader,
    IN  RvChar*                               strBuffer,
    IN  RvUint                                bufferLen,
    OUT RvUint*                               actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipSubscriptionStateHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Subscription-State header.

SYNTAX

```
RvStatus RvSipSubscriptionStateHeaderSetStrBadSyntax(
   IN RvSipSubscriptionStateHeaderHandle     hHeader,
   IN RvChar*                                strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

28

BODY FUNCTIONS

Body object functions enable you to construct, copy, encode, parse, access, add and change body parts and Content-Type. This section contains Body functions found in the *RvSipBody.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipBodyConstructInMsg()
- RvSipBodyConstructInBodyPart()
- RvSipBodyConstruct()
- RvSipBodyCopy()
- RvSipBodyEncode()
- RvSipBodyMultipartParse()

RvSipBodyConstructInMsg()

DESCRIPTION

Constructs a body object inside a given *message*.

SYNTAX

```
RvStatus RvSipBodyConstructInMsg(
    IN  RvSipMsgHandle     hSipMsg,
    OUT RvSipBodyHandle    *hBody);
```

PARAMETERS

hSipMsg

The handle of the message related to the new body object.

hBody

The handle of the newly constructed body object.

RETURN VALUES

Returns RvStatus.

RvSipBodyConstructInBodyPart()

DESCRIPTION

Constructs a body object inside a given body part object.

SYNTAX

```
RvStatus RvSipBodyConstructInBodyPart(
    IN  RvSipBodyPartHandle    hBodyPart,
    OUT RvSipBodyHandle        *hBody);
```

PARAMETERS

hBodyPart

The handle of the body part related to the new body object.

hBody

The handle of the newly constructed body object.

RETURN VALUES

Returns RvStatus.

RvSipBodyConstruct()

DESCRIPTION

Constructs and initializes a stand-alone body object. The body object is constructed on a given page taken from a specified pool. The handle of the new body object is returned.

SYNTAX

```
RvStatus RvSipBodyConstruct(
    IN  RvSipMsgMgrHandle    hMsgMgr,
    IN  HRPOOL               hPool,
    IN  HPAGE                hPage,
    OUT RvSipBodyHandle      *hBody);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hBody

The handle of the newly constructed body object.

RETURN VALUES

Returns RvStatus.

RvSipBodyCopy()

DESCRIPTION

Copies all parameters from a source body object to a destination body object. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipBodyCopy(
    INOUT RvSipBodyHandle    hDestination,
    IN    RvSipBodyHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination body object.

hSource

The handle of the source body object.

RETURN VALUES

Returns RvStatus.

RvSipBodyEncode()

DESCRIPTION

Encodes a body object to a textual body. The encoded body is placed on a page taken from a specified pool. To copy the encoded body from the page to a consecutive buffer, use RPOOL_CopyToExternal(). If this function returns RV_OK, the application must free the allocated page by using RPOOL_FreePage().

SYNTAX

```
RvStatus RvSipBodyEncode(
    IN  RvSipBodyHandle    hBody,
    IN  HRPOOL             hPool,
    OUT HPAGE*             hPage,
    OUT RvUint32*          length);
```

PARAMETERS

hBody

The handle of the body object.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded object.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The manner in which the body is encoded depends on the body part type, as follows:

- **Multipart body type**

  Each body part is encoded. The body parts are separated by a boundary delimiter line. The boundary delimiter line is created using the boundary parameter found in the Content-Type of the body object. If no boundary is defined for this body object, a unique boundary is generated and the boundary delimiter line is created accordingly.

- **Other body type**

  If the body type is other than multipart, it is copied to the given page.

RvSipBodyMultipartParse()

DESCRIPTION

Parses a SIP textual body of type multipart into a body object. The given body object must have a valid Content-Type header on its list of headers. The Content-Type header must contain the boundary that was used to create the boundary delimiter lines that separated the different parts of the body. The parsing will be done according to this boundary. See the Example below.

SYNTAX

```
RvStatus RvSipBodyMultipartParse(
    IN RvSipBodyHandle    hBody,
    IN RvChar             *strBody,
    IN RvUint             bufLength);
```

PARAMETERS

hBody

A handle of the body object.

strBody

The string of the body.

bufLength

The buffer length.

RETURN VALUES

Returns RvStatus.

EXAMPLE

The following multipart body is correctly parsed as follows:

`"boundary=unique-boundary-1"`

The multipart body must be a parameter of the Content-Type of the given body object.

```
unique-boundary-1
Content-Type: application/SDP; charset=ISO-10646
v=0
o=audet 2890844526 2890842807 5 IN IP4 134.177.64.4
s=SDP seminar
c=IN IP4 MG141.nortelnetworks.com
t= 2873397496 2873404696
m=audio 9092 RTP/AVP 0 3 4
unique-boundary-1
Content-type:application/QSIG; version=iso
08 02 55 55 05 04 02 90 90 18 03 a1 83 01
70 0a 89 31 34 30 38 34 39 35 35 30 37 32
unique-boundary-1
```

The body parts are separated into a list of body parts within the body object. The headers list of each body part is parsed within the appropriate body part object.

To Parse a body object you can either supply a string of your own, or you can use the body string within the body object, if the body object exists. The second option can be used when the body was received via the network. The body object will contain the body string. You can receive this string using RvSipBodyGetBodyStr(). To receive the length of this string in advance, use RvSipBodyGetBodyStrLength(). After you retrieve the body string, you can use RvSipBodyMultipartParse() to parse it.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipBodyGetContentType()
- RvSipBodySetContentType()
- RvSipBodyGetBodyStrLength()
- RvSipBodyGetBodyStr()
- RvSipBodySetBodyStr()
- RvSipBodyGetBodyPart()
- RvSipBodyPushBodyPart()
- RvSipBodyRemoveBodyPart()
- RvSipBodyGetRpoolString()
- RvSipBodySetRpoolString()

RvSipBodyGetContentType()

DESCRIPTION

Gets the Content-Type header from the body object.

SYNTAX

```
RvSipContentTypeHeaderHandle RvSipBodyGetContentType(
    IN RvSipBodyHandle    hBody);
```

PARAMETERS

hBody

The handle of the body object.

RETURN VALUES

Returns the Content-Type header handle, or NULL if there is no Content-Type for this body object.

RvSipBodySetContentType()

DESCRIPTION

Sets the Content-Type header in the body object.

SYNTAX

```
RvStatus RvSipBodySetContentType(
    IN RvSipBodyHandle                  hBody,
    IN RvSipContentTypeHeaderHandle     hContentType);
```

PARAMETERS

hBody

The handle of the body object.

hContentType

Content-Type header to be set in the body object.

RETURN VALUES

Returns RvStatus.

RvSipBodyGetBodyStrLength()

DESCRIPTION

Gets the length of the body string.

SYNTAX

```
RvUint32 RvSipBodyGetBodyStrLength(
    IN RvSipBodyHandle    hBody);
```

PARAMETERS

hBody

The handle of the body object.

RETURN VALUES

The length of the body string.

REMARKS

If the body is of type multipart, it might contain a list of body parts. The body string and the list of body parts may not represent the same body, since each of the body parts can be independently manipulated. You can use this function only when the body object does not contain a list of body parts. If the body object contains a list of body parts, this function will return zero. In this case, you can use RvSipBodyEncode().

RvSipBodyEncode() will encode the list of body parts and will return the length of the encoded string, which represents the length of the body string.

RvSipBodyGetBodyStr()

DESCRIPTION

Gets the string of the body.

SYNTAX

```
RvStatus RvSipBodyGetBodyStr(
    IN  RvSipBodyHandle    hBody,
    IN  RvChar             *strBuffer,
    IN  RvUint             bufferLen
    OUT RvUint*            actualLen);
```

PARAMETERS

hBody

The handle of the body object.

strBuffer

The buffer with which to fill the body string. The body string does not necessarily end with '\0'. The body string might contain '\0' as part of the body string.

bufferLen

The length of the buffer.

actualLen

The length of the body string. When the function returns, this parameter indicates the actual length of the buffer. This is also relevant when the buffer was insufficient.

REMARKS

If the body is of type multipart, it might contain a list of body parts. The body string and the list of body parts may not represent the same body, since each of the body parts can be independently manipulated. If the body object contains a list of body parts this function will return RV_ERROR_UNKNOWN. You can

only use this function when the body does not contain a list of body parts. If you would like to get the body string of a body object in the format of body parts list, use RvSipBodyEncode() and then the RPOOL_CopyToExternal() function.

RETURN VALUES

Returns RvStatus.

RvSipBodySetBodyStr()

DESCRIPTION

Sets a textual body to the body object. The textual body is copied and saved in the body object.

SYNTAX

```
RvStatus RvSipBodySetBodyStr(
    IN RvSipBodyHandle    hBody,
    IN RvChar             *rawBuffer,
    IN RvUint32           length);
```

PARAMETERS

hBody

The handle of the body object.

rawBuffer

The buffer to set to the body object.

length

The length of the buffer.

RETURN VALUES

Returns RvStatus.

RvSipBodyGetBodyPart()

DESCRIPTION

Gets a body part from the body parts list. The body object holds the body part objects in a sequential list.

SYNTAX

```
RvStatus RvSipBodyGetBodyPart(
    IN  RvSipBodyHandle        hBody,
    IN  RvSipListLocation      location,
    IN  RvSipBodyPartHandle    hRelative,
    OUT RvSipBodyPartHandle    *hBodyPart);
```

PARAMETERS

hBody

Handle of the body object.

location

The location in the list: next, previous, first or last.

hRelative

The handle of the current position in the list (a relative body part from the list). Supply this value if you choose next or previous in the location parameter.

hBodyPart

The handle of the requested body part.

RETURN VALUES

Returns RvStatus.

REMARKS

Use the location parameter to specify the location of the body part you wish to get. When using the next or prev locatopns, you must supply a relative body part. For example, in order to get the second body part from the list, you must first get the first body part. You should then ask for the next body part when the first body part serves as a relative header.

RvSipBodyPushBodyPart()

DESCRIPTION

Inserts a given body part into the body parts list based on a given location, which can be first, last, before or after a given relative element. The body part you supply is copied before it is inserted into the list. The function returns the handle of the body part object that was actually inserted into the list. This body part can be used in a following call to this function as the relative body part.

SYNTAX

```
RvStatus RvSipBodyPushBodyPart(
    IN  RvSipBodyHandle        hBody,
    IN  RvSipListLocation      location,
    IN  RvSipBodyPartHandle    hBodyPart,
    IN  RvSipBodyPartHandle    hRelative,
    OUT RvSipBodyPartHandle    *hNewBodyPart);
```

PARAMETERS

hBody

The handle of the body object.

location

The location in list, which can be next, previous, first or last.

hBodyPart

The handle of the body part to be pushed into the list.

hRelative

The handle of the current position in the list (a relative body part from the list). Supply this value if you chose next or previous in the location parameter.

hNewBodyPart

The handle of a copy of hBodyPart that was inserted to the body parts list. Can be used in further calls for this function as *hRelative*.

RETURN VALUES

Returns RvStatus.

RvSipBodyRemoveBodyPart()

DESCRIPTION

Removes a body part from the body parts list. You should supply to this function the handle of the body part you wish to remove.

SYNTAX

```
RvStatus RvSipBodyRemoveBodyPart(
    IN RvSipBodyHandle        hBody,
    IN RvSipBodyPartHandle    hBodyPart);
```

PARAMETERS

hBody

The handle of the body object.

hBodyPart

The handle of the body part to be removed.

RETURN VALUES

Returns RvStatus.

RvSipBodyGetRpoolString()

DESCRIPTION

Copies the body string from the body object into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipBodyGetRpoolString(
    IN    RvSipBodyHandle    hBody,
    INOUT RPOOL_Ptr          *pRpoolPtr);
```

PARAMETERS

hBody

The handle of the body object.

pRpoolPtr

A pointer to a location inside a RPOOL memory object. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set in the RPOOL memory.

RETURN VALUES

Returns RvStatus.

RvSipBodySetRpoolString()

DESCRIPTION

Sets a body string into a body object. The given string is located on an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipBodySetRpoolString(
    IN RvSipBodyHandle    hBody,
    IN RPOOL_Ptr          *pRpoolPtr,
    IN RvUint32           length);
```

PARAMETERS

hBody

The handle of the body object.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

length

The body length +1 for the terminating “\0” character.

RETURN VALUES

Returns RvStatus.

26

VIA HEADER FUNCTIONS

The Via header functions enable you to construct, copy, encode and parse Via headers and access Via header parameters. This section contains Via header functions found in the *RvSipViaHeader.h* header file.

The functions included in this section are:

- Control Functions
- Get and Set Functions

CONTROL FUNCTIONS

The Control functions are:

- RvSipViaHeaderConstructInMsg()
- RvSipViaHeaderConstruct()
- RvSipViaHeaderCopy()
- RvSipViaHeaderEncode()
- RvSipViaHeaderParse()
- RvSipViaHeaderParseValue()
- RvSipViaHeaderFix()

RvSipViaHeaderConstructInMsg()

DESCRIPTION

Constructs a Via header inside a given *message*. The header is kept in the header list of the message. You can choose to insert the header either at the head or tail of the list.

SYNTAX

```
RvStatus RvSipViaHeaderConstructInMsg(
    IN  RvSipMsgHandle           hSipMsg,
    IN  RvBool                   pushHeaderAtHead,
    OUT RvSipViaHeaderHandle*    hHeader);
```

PARAMETERS

hSipMsg

The handle of the *message*.

pushHeaderAtHead

A Boolean value indicating whether the header should be pushed to the head of the list (RV_TRUE) or to the tail (RV_FALSE).

hHeader

The handle of the newly constructed Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderConstruct()

DESCRIPTION

Constructs and initializes a stand-alone Via header. The header is constructed on a given page taken from a specified pool. The handle of the new header is returned.

SYNTAX

```
RvStatus RvSipViaHeaderConstruct(
    IN  RvSipMsgMgrHandle        hMsgMgr,
    IN  HRPOOL                   hPool,
    IN  HPAGE                    hPage,
    OUT RvSipViaHeaderHandle*    hHeader);
```

PARAMETERS

hMsgMgr

The handle of the *MessageMgr*.

hPool

The handle of the memory pool that the object will use.

hPage

The handle of the memory page that the object will use.

hHeader

The handle of the newly constructed Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderCopy()

DESCRIPTION

Copies all parameters from a source Via header to a destination Via header. You must construct the destination object before using this function.

SYNTAX

```
RvStatus RvSipViaHeaderCopy(
    INOUT RvSipViaHeaderHandle    hDestination,
    IN    RvSipViaHeaderHandle    hSource);
```

PARAMETERS

hDestination

The handle of the destination Via header.

hSource

The handle of the source Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderEncode()

DESCRIPTION

Encodes a Via header to a textual Via header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipViaHeaderEncode(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  HRPOOL                  hPool,
    OUT HPAGE*                  hPage,
    OUT RvUint32*               length);
```

PARAMETERS

hHeader

The handle of the Via header.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

REMARKS

The application must free the allocated page, using RPOOL_FreePage(). The allocated page must be freed only if this function returns RV_OK.

RvSipViaHeaderParse()

DESCRIPTION

Parses a SIP textual Via header into a Via header. All the textual parameters are placed inside the object. For example: “Via: SIP/2.0/UDP h.caller.com”.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipViaHeaderParse(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvChar*                 buffer);
```

PARAMETERS

hHeader

A handle of the Via header.

buffer

The buffer containing a textual Via header.

RETURN VALUES

Returns RvStatus.

REMARKS

You must construct a Via header before using this function.

RvSipViaHeaderParseValue()

DESCRIPTION

Parses a SIP textual Via header value into a Via header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipViaHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipViaHeaderParseValue(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvChar*                 buffer);
```

PARAMETERS

hHeader

The handle of the Via header.

buffer

The buffer containing a textual Via header value.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderFix()

DESCRIPTION

Fixes a Via header with bad-syntax information. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to fix the header by parsing a given correct header-value string to the supplied header. If parsing succeeds, this function places all parameters inside the object and removes the bad-syntax string. If parsing fails, the bad-syntax string in the header remains as it was.

SYNTAX

```
RvStatus RvSipViaHeaderFix(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvChar*                 pFixedBuffer);
```

PARAMETERS

hHeader

The handle of the header.

pFixedBuffer

The buffer containing a legal header value.

RETURN VALUES

Returns RvStatus.

GET AND SET FUNCTIONS

The Get and Set functions are:

- RvSipViaHeaderGetStringLength()
- RvSipViaHeaderGetTransport()
- RvSipViaHeaderGetStrTransport()
- RvSipViaHeaderSetTransport()
- RvSipViaHeaderGetHost()
- RvSipViaHeaderSetHost()
- RvSipViaHeaderGetPortNum()
- RvSipViaHeaderSetPortNum()
- RvSipViaHeaderGetTtlNum()
- RvSipViaHeaderSetTtlNum()
- RvSipViaHeaderGetMaddrParam()
- RvSipViaHeaderSetMaddrParam()
- RvSipViaHeaderGetReceivedParam()
- RvSipViaHeaderSetReceivedParam()
- RvSipViaHeaderGetRportParam()
- RvSipViaHeaderSetRportParam()
- RvSipViaHeaderGetBranchParam()
- RvSipViaHeaderSetBranchParam()
- RvSipViaHeaderGetHiddenParam()
- RvSipViaHeaderSetHiddenParam()
- RvSipViaHeaderGetOtherParams()
- RvSipViaHeaderSetOtherParams()
- RvSipViaHeaderGetRpoolString()
- RvSipViaHeaderSetRpoolString()
- RvSipViaHeaderGetStrBadSyntax()
- RvSipViaHeaderSetStrBadSyntax()
- RvSipViaHeaderGetCompParam()
- RvSipViaHeaderGetStrCompParam()
- RvSipViaHeaderSetCompParam()
- RvSipViaHeaderSetAliasParam()
- RvSipViaHeaderGetAliasParam()
- RvSipViaHeaderSetCompactForm()

- RvSipViaHeaderGetCompactForm()
- RvSipViaHeaderGetSigCompIdParam()
- RvSipViaHeaderSetSigCompIdParam()

RvSipViaHeaderGetStringLength()

DESCRIPTION

Some of the Via header parameters are kept in a string format, for example, the host parameter. In order to get such a parameter from the Via header, your application should supply an adequate buffer to where the string will be copied. This function provides you with the length of the string to enable you to allocate an appropriate buffer size before calling the Get function.

SYNTAX

```
RvUint RvSipViaHeaderGetStringLength(
    IN RvSipViaHeaderHandle        hHeader,
    IN RvSipViaHeaderStringName    stringName);
```

PARAMETERS

hHeader

The handle of the Via header.

stringName

The enumeration of the string name for which you require the length.

RETURN VALUES

Returns the length of the specified string.

RvSipViaHeaderGetTransport()

DESCRIPTION

Gets the transport protocol enumeration from the Via header.

SYNTAX

```
RvSipTransport RvSipViaHeaderGetTransport(
    IN RvSipViaHeaderHandle    hViaHeader);
```

PARAMETERS

hViaHeader

The handle of the Via header.

RETURN VALUES

Returns the transport protocol enumeration of the Via header.

REMARK

If this function returns RVSIP_TRANSPORT_OTHER, call RvSipViaHeaderGetStrTransport() to get the transport protocol in a string format.

RvSipViaHeaderGetStrTransport()

DESCRIPTION

Copies the transport protocol string from the Via header into a given buffer. Use this function if RvSipViaHeaderGetTransport() returns RVSIP_TRANSPORT_OTHER. If the *bufferLen* size is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetStrTransport(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the Via header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetTransport()

DESCRIPTION

Sets the transport type the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetTransport(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvSipTransport          eTransport,
    IN RvChar*                 strTransport);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

eTransport

The enumeration transport protocol to set in the object.

strTransport

You can use this parameter only if the *eTransport* parameter is set to RVSIP_TRANSPORT_OTHER. In this case you can supply the transport protocol as a string.

RETURN VALUES

Returns RvStatus.

REMARKS

If *eTransport* is set to RVSIP_TRANSPORT_OTHER *strTransport* is copied to the header. Otherwise, *strTransport* is ignored.

RvSipViaHeaderGetHost()

DESCRIPTION

Copies the host name, send-by parameter, from the Via header into a given buffer. If the *bufferLen* size is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetHost(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetHost()

DESCRIPTION

Sets the host parameter in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetHost(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvChar*                 strHost);
```

PARAMETERS

hViaHeader

The handle of the Via header.

strHost

The host to be set in the object. If Null is supplied, the existing host string is removed from the object.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetPortNum()

DESCRIPTION

Gets the host port from the Via header.

SYNTAX

```
RvInt32 RvSipViaHeaderGetPortNum(
    IN RvSipViaHeaderHandle    hViaHeader);
```

PARAMETERS

hViaHeader

The handle of the Via header.

RETURN VALUES

Returns the port number, or UNDEFINED if the port number does not exist.

RvSipViaHeaderSetPortNum()

DESCRIPTION

Sets the host port number of the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetPortNum(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvInt32                 portNum);
```

PARAMETERS

hViaHeader

The handle of the Via header.

portNum

The port number value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetTtlNum()

DESCRIPTION

Gets the ttl from the Via header.

SYNTAX

```
RvInt16 RvSipViaHeaderGetTtlNum(
    IN RvSipViaHeaderHandle    hViaHeader);
```

PARAMETERS

hViaHeader

The handle of the Via header.

RETURN VALUES

Returns the ttl number, or UNDEFINED if the ttl does not exist.

RvSipViaHeaderSetTtlNum()

DESCRIPTION

Sets the ttl parameter in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetTtlNum(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvInt16                 ttlNum);
```

PARAMETERS

hViaHeader

The handle of the Via header.

ttlNum

The ttl value to be set in the object.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetMaddrParam()

DESCRIPTION

Copies the maddr parameter of the Via header into a given buffer. If the *bufferLen* size is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetMaddrParam(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

Buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetMaddrParam()

DESCRIPTION

Sets the maddr string parameter in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetMaddrParam(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvChar                  *strMaddrParam);
```

PARAMETERS

hViaHeader

The handle of the Via header.

strMaddrParam

The maddr parameter to be set in the object. If NULL is supplied, the existing maddr parameter string is removed from the object.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetReceivedParam()

DESCRIPTION

Copies the received parameter of the Via header into a given buffer. If the *bufferLen* size is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetReceivedParam(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetReceivedParam()

DESCRIPTION

Sets the value of the received string parameter in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetReceivedParam(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvChar*                 strReceivedParam);
```

PARAMETERS

hViaHeader

The handle of the Via header.

strReceivedParam

The received parameter to be set in the object. If NULL is supplied, the existing received parameter string is removed from the object.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetRportParam()

DESCRIPTION

Gets the rport parameter from the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderGetRportParam(
    IN  RvSipViaHeaderHandle    hSipViaHeader,
    OUT RvInt32*                rportParam,
    OUT RvBool*                 bUseRport);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

rportParam

The rport value (number), or UNDEFINED if the rport parameter is empty.

bUseRport

Indicates whether or not the rport parameter exists in the header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetRportParam()

DESCRIPTION

Sets the rport parameter value in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetRportParam(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvInt32                 rportParam,
    IN RvBool                  useRport);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

rportParam

The rport number to be set in the Via header (UNDEFINED is possible for an empty rport parameter).

useRport

Indicates whether or not the rport should be added to the Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetBranchParam()

DESCRIPTION

Copies the branch parameter from the Via header into a given buffer. If the *bufferLen* size is adequate, the function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetBranchParam(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetBranchParam()

DESCRIPTION

Sets the branch parameter in the Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetBranchParam(
    IN RvSipViaHeaderHandle    hViaHeader,
    IN RvChar*                 strBranchParam);
```

PARAMETERS

hViaHeader

The handle of the Via header.

strBranchParam

The branch parameter to be set in the object. If NULL is supplied, the existing branch string is removed from the Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetHiddenParam()

DESCRIPTION

Gets the hidden parameter from the Via header.

SYNTAX

```
RvBool RvSipViaHeaderGetHiddenParam(
    IN RvSipViaHeaderHandle    hSipViaHeader);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

RETURN VALUES

Returns the hidden parameter value. If the hidden parameter exists in the Via header, RV_TRUE will be returned. Otherwise returns RV_FALSE.

RvSipViaHeaderSetHiddenParam()

DESCRIPTION

Sets the hidden parameter in the Via header. If the hidden parameter exists in the Via header, RV_TRUE will be set, Otherwise, RV_FALSE will be set.

SYNTAX

```
RvStatus RvSipViaHeaderSetHiddenParam(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvBool                  hiddenValue);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

hiddenValue

The hidden parameter value to set.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetOtherParams()

DESCRIPTION

Copies the Via header other-params parameter of the Via header into a given buffer. Not all the Via header parameters have separated parameters in the Via header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”. If the *bufferLen* is adequate, the function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetOtherParams(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetOtherParams()

DESCRIPTION

Sets the other-params parameter in the Via header. Not all the Via header parameters have separated parameters in the Via header. Parameters with no specific parameters are referred to as “other-params”. They are kept in the object in one concatenated string in the following form: “name=value;name=value...”.

SYNTAX

```
RvStatus RvSipViaHeaderSetOtherParams(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvChar*                 strViaParams);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

strViaParams

The Via Params string to be set in the Via header. If NULL is supplied, the existing other-params parameter is removed from the header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetRpoolString()

DESCRIPTION

Copies a string parameter from the Via header into a given page from a specified pool. The copied string is not consecutive.

SYNTAX

```
RvStatus RvSipViaHeaderGetRpoolString(
    IN    RvSipViaHeaderHandle        hSipViaHeader,
    IN    RvSipViaHeaderStringName    eStringName,
    INOUT RPOOL_Ptr                   *pRpoolPtr);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

eStringName

The string the user wishes to get.

pRpoolPtr

A pointer to a location inside an RPOOL. You need to supply only the pool and page. The offset where the returned string was located will be returned as an output parameter. If the function set the returned offset to UNDEFINED, this means that the parameter was not set to the Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetRpoolString()

DESCRIPTION

Sets a string into a specified parameter in the Via header. The given string is located on an RPOOL memory and is not consecutive.

SYNTAX

```
RvStatus RvSipViaHeaderSetRpoolString(
    IN RvSipViaHeaderHandle        hSipViaHeader,
    IN RvSipViaHeaderStringName    eStringName,
    IN RPOOL_Ptr                   *pRpoolPtr);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

eStringName

The string the user wishes to set.

pRpoolPtr

A pointer to a location inside an RPOOL where the new string is located.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetStrBadSyntax()

DESCRIPTION

Copies the bad-syntax string from the header into a given buffer. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. You use this function to retrieve the bad-syntax string. If the value of *bufferLen* is adequate, this function copies the requested parameter into *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and *actualLen* contains the required buffer length. Use this function in the RvSipTransportBadSyntaxMsgEv() callback implementation if the message contains a bad Via header, and you wish to see the header-value.

SYNTAX

```
RvStatus RvSipViaHeaderGetStrBadSyntax(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer with which to fill the bad-syntax string.

bufferLen

The length of the buffer.

actualLen

The length of the bad-syntax string + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetStrBadSyntax()

DESCRIPTION

Sets a bad-syntax string in the object. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. By using this function you can create a header with a bad-syntax string. Setting a bad-syntax string to the header will mark the header as an invalid syntax header. You can use his function when you want to send an illegal Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetStrBadSyntax(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvChar*                 strBadSyntax);
```

PARAMETERS

hHeader

The handle of the header.

strBadSyntax

The bad-syntax string.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetCompParam()

DESCRIPTION

Gets the comp enumeration from the Via header. If this function returns RVSIP_SIGCOMP_OTHER, call RvSipViaHeaderGetStrCompParam() to get the comp parameter in string format.

SYNTAX

```
RvSipCompType RvSipViaHeaderGetCompParam(
    IN RvSipViaHeaderHandle    hSipViaHeader);
```

PARAMETERS

hViaHeader

The handle of the Via header.

RETURN VALUES

Returns the comp enumeration of the Via header.

RvSipViaHeaderGetStrCompParam()

DESCRIPTION

Copies the comp parameter from the Via header into a given buffer. If the *bufferLen* size is adequate, this function copies the parameter into the *strBuffer*. Otherwise, this function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetStrCompParam(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle of the header.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetCompParam()

DESCRIPTION

Sets the compression type in the Via header. You can use *strComp* parameter only if the *eComp* parameter is set to RVSIP_COMP_OTHER. In this case, you can supply the compression type as a string. If *eComp* is set to RVSIP_COMP_OTHER, *strComp* is copied to the header. Otherwise, *strComp* is ignored.

SYNTAX

```
RvStatus RvSipViaHeaderSetCompParam(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvSipCompType           eComp,
    IN RvChar*                 strComp);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

eComp

The enumeration transport protocol to set in the object.

strComp

You can use this parameter only if the *eComp* parameter is set to RVSIP_COMP_OTHER. In this case, you can supply the compression type as a string.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetAliasParam()

DESCRIPTION

Sets the alias parameter in a Via header.

SYNTAX

```
RvStatus RvSipViaHeaderSetAliasParam(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvBool                  bAlias);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

bAlias

Indicates whether or not the alias should be added to the Via header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetAliasParam()

DESCRIPTION

Gets the alias parameter from the Via header.

SYNTAX

```
RvBool RvSipViaHeaderGetAliasParam(
    IN RvSipViaHeaderHandle    hSipViaHeader);
```

PARAMETERS

hSipViaHeader

The handle of the Via header.

RETURN VALUES

Returns RV_TRUE if the alias parameter exists. Otherwise returns RV_FALSE.

RvSipViaHeaderSetCompactForm()

DESCRIPTION

Instructs the header to use the compact header name when the header is encoded.

SYNTAX

```
RvStatus RvSipViaHeaderSetCompactForm(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvBool                  bIsCompact);
```

PARAMETERS

hHeader

The handle of the Via header.

bIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetCompactForm()

DESCRIPTION

Gets the compact form flag of the header.

SYNTAX

```
RvStatus RvSipViaHeaderGetCompactForm(
    IN RvSipViaHeaderHandle    hHeader,
    IN RvBool                  *pbIsCompact);
```

PARAMETERS

hHeader

The handle of the Via header.

pbIsCompact

Specifies whether or not the header is a compact form header.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderGetSigCompIdParam()

DESCRIPTION

Copies the SigCompId parameter from the Via header object into a given buffer. If the *bufferLen* size is adequate, this function copies the parameter into the *strBuffer*. Otherwise, the function returns RV_ERROR_INSUFFICIENT_BUFFER and the *actualLen* parameter contains the required buffer length.

SYNTAX

```
RvStatus RvSipViaHeaderGetSigCompIdParam(
    IN  RvSipViaHeaderHandle    hHeader,
    IN  RvChar*                 strBuffer,
    IN  RvUint                  bufferLen,
    OUT RvUint*                 actualLen);
```

PARAMETERS

hHeader

The handle to the header object.

strBuffer

The buffer to fill with the requested parameter.

bufferLen

The length of the buffer.

actualLen

The length of the requested parameter + 1, to include a NULL value at the end of the parameter.

RETURN VALUES

Returns RvStatus.

RvSipViaHeaderSetSigCompIdParam()

DESCRIPTION

Sets the sigcomp-id parameter in the Via header object.

SYNTAX

```
RvStatus RvSipViaHeaderSetSigCompIdParam(
    IN RvSipViaHeaderHandle    hSipViaHeader,
    IN RvChar*                 strSigCompId);
```

PARAMETERS

hSipViaHeader

The handle to the Via header object.

strSigCompId

The sigcomp-id parameter to set.

RETURN VALUES

Returns RvStatus.

Get and Set Functions
RvSipViaHeaderSetSigCompIdParam()

27

FUNCTIONS FOR ALL HEADERS

These functions allow you to encode, parse and parseValue every header (without using the header-specific functions). The functions receive the header type as a parameter and, according to this parameter, call the header specific-function. This section contains functions that can be used for all headers, and are found in the *RvSipHeader.h* header file.

The functions included in this section are:

CONTROL FUNCTIONS

- RvSipHeaderEncode()
- RvSipHeaderParse()
- RvSipHeaderParseValue()
- RvSipHeaderIsBadSyntax()

CONTROL FUNCTIONS

RvSipHeaderEncode()

DESCRIPTION

Encodes a header of any type to a textual header. The textual header is placed on a page taken from a specified pool. To copy the textual header from the page to a consecutive buffer, use RPOOL_CopyToExternal().

SYNTAX

```
RvStatus RvSipHeaderEncode(
    IN  void*              pHeader,
    IN  RvSipHeaderType    eType,
    IN  HRPOOL             hPool,
    OUT HPAGE*             hPage,
    OUT RvUint32*          length);
```

PARAMETERS

pHeader

The handle of the header structure.

eType

The type of the header to encode.

hPool

The handle of the specified memory pool.

hPage

The memory page allocated to contain the encoded header.

length

The length of the encoded information.

RETURN VALUES

Returns RvStatus.

RvSipHeaderParse()

DESCRIPTION

Parses a SIP textual header, of any type, into a header. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

SYNTAX

```
RvStatus RvSipHeaderParse(
    IN void*              hHeader,
    IN RvSipHeaderType    eType,
    IN RvChar*            buffer);
```

PARAMETERS

hHeader

The handle of the header.

eType

The type of the header to parse.

buffer

The buffer containing a textual header.

RETURN VALUES

Returns RvStatus.

RvSipHeaderParseValue()

DESCRIPTION

Parses a SIP textual header value, of any header type, into a header. A SIP header has the following grammar: “header-name:header-value”. This function takes the header-value part as a parameter and parses it into the supplied object. All the textual parameters are placed inside the object.

Note Before performing the parse operation, the SIP Stack resets all the header parameters. Therefore, if the parse function fails, you will not be able to get the previous header parameter values.

Note Use the RvSipHeaderParse() function to parse strings that also include the header-name.

SYNTAX

```
RvStatus RvSipHeaderParseValue(
    IN void*              hHeader,
    IN RvSipHeaderType    eType,
    IN RvChar*            buffer);
```

PARAMETERS

hHeader

The handle of the header.

eType

Type of the header to parse.

buffer

The buffer containing the textual header.

RETURN VALUES

Returns RvStatus.

RvSipHeaderIsBadSyntax()

DESCRIPTION

Checks if the given header is a bad-syntax header or not. A SIP header has the following grammar: “header-name:header-value”. When a header contains a syntax error, the header-value is kept as a separate bad-syntax string. Use this function to check whether or not this header contains a bad-syntax string.

SYNTAX

```
RvBool RvSipHeaderIsBadSyntax(
    IN void*              pHeader,
    IN RvSipHeaderType    eHeaderType);
```

PARAMETERS

pHeader

A handle of the header.

eHeaderType

The type of given header.

RETURN VALUES

Returns RV_TRUE the if the header contains a bad-syntax string or RV_FALSE if header does not contain a bad-syntax string. If *pHeader* is NULL, the function returns RV_FALSE.

Type Definitions and Status Codes

This part includes the following sections:

- Message Type Definitions

9

MESSAGE TYPE DEFINITIONS

This section includes The SIP Stack Message type definitions defined in the *RvSipMsgTypes.h* and *RvSipXXXHeader.h* header files.

This section includes:

- Handle Type Definitions
- String Name Type Definitions
- Message API Type Definitions

HANDLE TYPE DEFINITIONS

The Handle type definitions are:

- RvSipAcceptHeaderHandle
- RvSipAcceptEncodingHeaderHandle
- RvSipAcceptLanguageHeaderHandle
- RvSipAddressHandle
- RvSipAllowEventsHeaderHandle
- RvSipAllowHeaderHandle
- RvSipAuthenticationHeaderHandle
- RvSipAuthenticationInfoHeaderHandle
- RvSipAuthorizationHeaderHandle
- RvSipBodyHandle
- RvSipBodyPartHandle
- RvSipContactHeaderHandle
- RvSipContentDispositionHeaderHandle
- RvSipContentIDHeaderHandle
- RvSipContentTypeHeaderHandle
- RvSipCSeqHeaderHandle
- RvSipDateHeaderHandle
- RvSipEventHeaderHandle
- RvSipExpiresHeaderHandle
- RvSipHeadersListElemHandle
- RvSipInfoHeaderHandle
- RvSipMinSEHeaderHandle
- RvSipMsgHandle
- RvSipMsgMgrHandle
- RvSipOtherHeaderHandle
- RvSipPartyHeaderHandle
- RvSipRAckHeaderHandle
- RvSipReasonHeaderHandle
- RvSipReferredByHeaderHandle
- RvSipReferToHeaderHandle
- RvSipReplacesHeaderHandle
- RvSipReplyToHeaderHandle

- RvSipRetryAfterHeaderHandle
- RvSipRouteHopHeaderHandle
- RvSipRSeqHeaderHandle
- RvSipSessionExpiresHeaderHandle
- RvSipSubscriptionStateHeaderHandle
- RvSipTimestampHeaderHandle
- RvSipViaHeaderHandle
- RvSipWarningHeaderHandle

RvSipAcceptHeaderHandle

DESCRIPTION

The handle of the Accept header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAcceptHeaderHandle);
```

RvSipAcceptEncodingHeaderHandle

DESCRIPTION

The handle of the Accept-Encoding header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAcceptEncodingHeaderHandle);
```

RvSipAcceptLanguageHeaderHandle

DESCRIPTION

The handle of the Accept-Language header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAcceptLanguageHeaderHandle);
```

RvSipAddressHandle

DESCRIPTION

The handle of the SIP address object.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAddressHandle);
```

RvSipAllowEventsHeaderHandle

DESCRIPTION

The handle of the Allow-Events header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAllowEventsHeaderHandle);
```

RvSipAllowHeaderHandle

DESCRIPTION

The handle of the Allow header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAllowHeaderHandle);
```

RvSipAuthenticationHeaderHandle

DESCRIPTION

The handle of the Authentication header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAuthenticationHeaderHandle);
```

RvSipAuthenticationInfoHeaderHandle

DESCRIPTION

The handle of the Authentication-Info header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAuthenticationInfoHeaderHandle);
```

RvSipAuthorizationHeaderHandle

DESCRIPTION

The handle of the Authorization header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipAuthorizationHeaderHandle);
```

RvSipBodyHandle

DESCRIPTION

The handle of the message body object.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipBodyHandle);
```

RvSipBodyPartHandle

DESCRIPTION

The handle of the message body part object.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipBodyPartHandle);
```

RvSipContentDispositionHeaderHandle

DESCRIPTION

The handle of the Content-Disposition header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipContentDispositionHeaderHandle);
```

RvSipContentIDHeaderHandle

DESCRIPTION

The handle of the Content-ID header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipContentIDHeaderHandle);
```

RvSipContactHeaderHandle

DESCRIPTION

The handle of the Contact header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipContactHeaderHandle);
```

RvSipContentTypeHeaderHandle

DESCRIPTION

The handle of the Content-Type header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipContentTypeHeaderHandle);
```

RvSipCSeqHeaderHandle

DESCRIPTION

The handle of the CSeq header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipCSeqHeaderHandle);
```

RvSipDateHeaderHandle

DESCRIPTION

The handle of the Date header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipDateHeaderHandle);
```

RvSipEventHeaderHandle

DESCRIPTION

The handle of the Event header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipEventHeaderHandle);
```

RvSipExpiresHeaderHandle

DESCRIPTION

The handle of the expires header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipExpiresHeaderHandle);
```

RvSipHeadersListElemHandle

DESCRIPTION

The handle of the position of one element in the header list. This handle is used for getting and setting headers in the header list.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipHeaderListElemHandle);
```

RvSipInfoHeaderHandle

DESCRIPTION

The handle of the Info header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipInfoHeaderHandle);
```

RvSipMinSEHeaderHandle

DESCRIPTION

The handle of the min-se header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipMinSEHeaderHandle);
```

RvSipMsgHandle

DESCRIPTION

The handle of the *message*.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipMsgHandle);
```

RvSipMsgMgrHandle

DESCRIPTION

The handle of the *MessageMgr*.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipMsgMgrHandle);
```

RvSipOtherHeaderHandle

DESCRIPTION

The handle of the Other header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipOtherHeaderHandle);
```

RvSipPartyHeaderHandle

DESCRIPTION

The handle of the Party header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipPartyHeaderHandle);
```

RvSipRAckHeaderHandle

DESCRIPTION

The handle of the RAck header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipRAckHeaderHandle);
```

RvSipReasonHeaderHandle

DESCRIPTION

The handle of the Reason header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipReasonHeaderHandle);
```

RvSipReferredByHeaderHandle

DESCRIPTION

The handle of the Referred-By header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipReferredByHeaderHandle);
```

RvSipReferToHeaderHandle

DESCRIPTION

The handle of the Refer-To header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipReferToHeaderHandle);
```

RvSipReplacesHeaderHandle

DESCRIPTION

The handle of the Replaces header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipReplacesHeaderHandle);
```

RvSipReplyToHeaderHandle

DESCRIPTION

The handle of the Reply-To header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipReplyToHeaderHandle);
```

RvSipRetryAfterHeaderHandle

DESCRIPTION

The handle of the Retry-After header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipRetryAfterHeaderHandle);
```

RvSipRouteHopHeaderHandle

DESCRIPTION

The handle of the Route-Hop header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipRouteHopHeaderHandle);
```

RvSipRSeqHeaderHandle

DESCRIPTION

The handle of the RSeq header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipRSeqHeaderHandle);
```

RvSipSessionExpiresHeaderHandle

DESCRIPTION

The handle of the Session-Expires header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipSessionExpiresHeaderHandle);
```

RvSipSubscriptionStateHeaderHandle

DESCRIPTION

The handle of the Subscription-State header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipSubscriptionStateHeaderHandle);
```

RvSipTimestampHeaderHandle

DESCRIPTION

The handle of the Timestamp header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipTimestampHeaderHandle);
```

RvSipViaHeaderHandle

DESCRIPTION

The handle of the Via header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipViaHeaderHandle);
```

RvSipWarningHeaderHandle

DESCRIPTION

The handle of the Warning header.

SYNTAX

```
RV_DECLARE_HANDLE(RvSipWarningHeaderHandle);
```

STRING NAME TYPE DEFINITIONS

The String Name type definitions are:

- RvSipAddressStringName
- RvSipAllowEventsHeaderStringName
- RvSipAllowHeaderStringName
- RvSipAuthenticationHeaderStringName
- RvSipAuthorizationHeaderStringName
- RvSipContactHeaderStringName
- RvSipContentDispositionHeaderStringName
- RvSipContentTypeHeaderStringName
- RvSipContentIDHeaderStringName
- RvSipContentIDHeaderFieldName
- RvSipCSeqHeaderStringName
- RvSipDateHeaderStringName
- RvSipEventHeaderStringName
- RvSipExpiresHeaderStringName
- RvSipMessageStringName
- RvSipMinSEHeaderStringName
- RvSipOtherHeaderStringName
- RvSipPartyHeaderStringName
- RvSipRAckHeaderStringName
- RvSipReferredByHeaderStringName
- RvSipReferToHeaderStringName
- RvSipReplacesHeaderStringName
- RvSipRetryAfterHeaderStringName
- RvSipRouteHopHeaderStringName
- RvSipRSeqHeaderStringName
- RvSipSessionExpiresHeaderStringName
- RvSipSubscriptionStateHeaderStringName
- RvSipViaHeaderStringName
- RvSipReplacesHeaderStringName
- RvSipAuthenticationInfoHeaderStringName

RvSipAddressStringName

DESCRIPTION

Defines all *address* parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipAddress.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_ADDRESS_ABS_URI_SCHEME,
    RVSIP_ADDRESS_ABS_URI_IDENTIFIER,
    RVSIP_ADDRESS_USER,
    RVSIP_ADDRESS_HOST,
    RVSIP_ADDRESS_TRANSPORT,
    RVSIP_ADDRESS_MADDR_PARAM,
    RVSIP_ADDRESS_USER_PARAM,
    RVSIP_ADDRESS_COMP_PARAM,
    RVSIP_ADDRESS_SIGCOMPID_PARAM,
    RVSIP_ADDRESS_TOKENIZED_BY_PARAM,
    RVSIP_ADDRESS_URL_OTHER_PARAMS,
    RVSIP_ADDRESS_HEADERS,
    RVSIP_ADDRESS_METHOD,
    RVSIP_ADDRESS_TEL_URI_NUMBER,
    RVSIP_ADDRESS_TEL_URI_EXTENSION,
    RVSIP_ADDRESS_TEL_URI_CONTEXT,
    RVSIP_ADDRESS_TEL_URI_ISDN_SUBS,
    RVSIP_ADDRESS_TEL_URI_POST_DIAL,
    RVSIP_ADDRESS_TEL_URI_OTHER_PARAMS,
    RVSIP_ADDRESS_DISPLAY_NAME,
    RVSIP_ADDRESS_CPC,
    RVSIP_ADDRESS_GR,
    RVSIP_ADDRESS_TEL_URI_CPC,
    RVSIP_ADDRESS_TEL_URI_RN,
    RVSIP_ADDRESS_TEL_URI_RN_CONTEXT,
    RVSIP_ADDRESS_TEL_URI_CIC,
    RVSIP_ADDRESS_TEL_URI_CIC_CONTEXT,
```

```
    RVSIP_ADDRESS_DIAMETER_URI_HOST,
    RVSIP_ADDRESS_DIAMETER_URI_TRANSPORT,
    RVSIP_ADDRESS_DIAMETER_URI_OTHER_PARAMS
}RvSipAddressStringName;
```

RvSipAllowEventsHeaderStringName

DESCRIPTION

Defines all Allow-Events header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipAllowEventsHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_ALLOW_EVENTS_EVENT_PACKAGE,
    RVSIP_ALLOW_EVENTS_EVENT_TEMPLATE,
    RVSIP_ALLOW_EVENTS_BAD_SYNTAX
}RvSipAllowEventsHeaderStringName;
```

RvSipAllowHeaderStringName

DESCRIPTION

Defines all Allow header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipAllowHeader.h* header file.

SYNTAX

```
typedef enum{
     RVSIP_ALLOW_METHOD,
     RVSIP_ALLOW_BAD_SYNTAX
}RvSipAllowHeaderStringName;
```

RvSipAuthenticationHeaderStringName

DESCRIPTION

Defines all Authentication header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipAuthenticationHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_AUTHENTICATION_AUTHSCHEME,
    RVSIP_AUTHENTICATION_REALM,
    RVSIP_AUTHENTICATION_DOMAIN,
    RVSIP_AUTHENTICATION_NONCE,
    RVSIP_AUTHENTICATION_OPAQUE,
    RVSIP_AUTHENTICATION_ALGORITHM,
    RVSIP_AUTHENTICATION_QOP,
    RVSIP_AUTHENTICATION_OTHER_PARAMS,
    RVSIP_AUTHENTICATION_BAD_SYNTAX
}RvSipAuthenticationHeaderStringName;
```

RvSipAuthorizationHeaderStringName

DESCRIPTION

Defines all Authorization header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipAuthorizationHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_AUTHORIZATION_AUTHSCHEME,
    RVSIP_AUTHORIZATION_USERNAME,
    RVSIP_AUTHORIZATION_REALM,
    RVSIP_AUTHORIZATION_NONCE,
    RVSIP_AUTHORIZATION_CNONCE,
    RVSIP_AUTHORIZATION_URI,
    RVSIP_AUTHORIZATION_OPAQUE,
    RVSIP_AUTHORIZATION_QOP,
    RVSIP_AUTHORIZATION_RESPONSE,
    RVSIP_AUTHORIZATION_ALGORITHM,
    RVSIP_AUTHORIZATION_OTHER_PARAMS,
    RVSIP_AUTHORIZATION_BAD_SYNTAX
}RvSipAuthorizationHeaderStringName;
```

RvSipContactHeaderStringName

DESCRIPTION

Defines all Contact header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipContactHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_CONTACT_DISPLAYNAME,
    RVSIP_CONTACT_OTHER_PARAMS,
    RVSIP_CONTACT_TEMP_GRUU,
    RVSIP_CONTACT_PUB_GRUU,
    RVSIP_CONTACT_QVAL,
    RVSIP_CONTACT_AUDIO,
    RVSIP_CONTACT_AUTOMATA,
    RVSIP_CONTACT_CLASS,
    RVSIP_CONTACT_DUPLEX,
    RVSIP_CONTACT_DATA,
    RVSIP_CONTACT_CONTROL,
    RVSIP_CONTACT_MOBILITY,
    RVSIP_CONTACT_DESCRIPTION,
    RVSIP_CONTACT_EVENTS,
    RVSIP_CONTACT_PRIORITY,
    RVSIP_CONTACT_METHODS,
    RVSIP_CONTACT_SCHEMES,
    RVSIP_CONTACT_APPLICATION,
    RVSIP_CONTACT_VIDEO,
    RVSIP_CONTACT_LANGUAGE,
    RVSIP_CONTACT_TYPE,
    RVSIP_CONTACT_ISFOCUS,
    RVSIP_CONTACT_ACTOR,
    RVSIP_CONTACT_TEXT,
    RVSIP_CONTACT_EXTENSIONS,
    RVSIP_CONTACT_SIP_INSTANCE,
    RVSIP_CONTACT_BAD_SYNTAX
```

```
}RvSipContactHeaderStringName
```

RvSipContentDispositionHeaderStringName

DESCRIPTION

Defines all Content-Disposition header parameters that are kept in the object in a string format.This enumeration is defined in the *RvSipContentDispositionHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_CONTENT_DISPOSITION_TYPE,
    RVSIP_CONTENT_DISPOSITION_HANDLING,
    RVSIP_CONTENT_DISPOSITION_OTHER_PARAMS,
    RVSIP_CONTENT_DISPOSITION_BAD_SYNTAX
}RvSipContentDispositionHeaderStringName;
```

RvSipContentTypeHeaderStringName

DESCRIPTION

Defines all Content-Type header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipContentTypeHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_CONTENT_TYPE_MEDIATYPE,
    RVSIP_CONTENT_TYPE_MEDIASUBTYPE,
    RVSIP_CONTENT_TYPE_BOUNDARY,
    RVSIP_CONTENT_TYPE_VERSION,
    RVSIP_CONTENT_TYPE_BASE,
    RVSIP_CONTENT_TYPE_OTHER_PARAMS,
    RVSIP_CONTENT_TYPE_BAD_SYNTAX,
    RVSIP_CONTENT_TYPE_PARAM_MEDIATYPE,
    RVSIP_CONTENT_TYPE_PARAM_MEDIASUBTYPE
}RvSipContentTypeHeaderStringName;
```

RvSipContentIDHeaderStringName

DESCRIPTION

Defines all Content-ID header object fields that are kept in the object in a string format.

SYNTAX

```
typedef enum{
    RVSIP_CONTENT_ID_BAD_SYNTAX
}RvSipContentIDHeaderStringName;
```

RvSipContentIDHeaderFieldName

DESCRIPTION

Defines all the Content-ID header fields. It is used for getting and setting values via the RvSipHeader interface.

SYNTAX

```
typedef enum {
    RVSIP_CONTENT_ID_FIELD_BAD_SYNTAX = 0
}RvSipContentIDHeaderFieldName;
```

RvSipCSeqHeaderStringName

DESCRIPTION

Defines all CSeq header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipCSeqHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_CSEQ_METHOD,
    RVSIP_CSEQ_BAD_SYNTAX
}RvSipCSeqHeaderStringName;
```

RvSipDateHeaderStringName

DESCRIPTION

Defines all Date header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipDateHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_DATE_BAD_SYNTAX
}RvSipDateHeaderStringName;
```

RvSipEventHeaderStringName

DESCRIPTION

Defines all Event header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipEventHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_EVENT_PACKAGE,
    RVSIP_EVENT_TEMPLATE,
    RVSIP_EVENT_ID,
    RVSIP_EVENT_PARAM,
    RVSIP_EVENT_BAD_SYNTAX
}RvSipEventHeaderStringName;
```

RvSipExpiresHeaderStringName

DEFINITION

Defines all Expires header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipExpiresHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_EXPIRES_BAD_SYNTAX
}RvSipExpiresHeaderStringName;
```

RvSipMessageStringName

DESCRIPTION

Defines all *message* parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipMsg.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_MSG_REQUEST_METHOD,
    RVSIP_MSG_RESPONSE_PHRASE,
    RVSIP_MSG_CALL_ID,
    RVSIP_MSG_CONTENT_TYPE,
    RVSIP_MSG_BODY,
    RVSIP_MSG_BAD_SYNTAX_START_LINE
}RvSipMessageStringName;
```

RvSipMinSEHeaderStringName

DESCRIPTION

Defines all Min-SE header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipMinSEHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_MINSE_OTHER_PARAMS,
    RVSIP_MINSE_BAD_SYNTAX
} RvSipMinSEHeaderStringName;
```

RvSipOtherHeaderStringName

DESCRIPTION

Defines all Other header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipOtherHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_OTHER_HEADER_NAME,
    RVSIP_OTHER_HEADER_VALUE,
    RVSIP_OTHER_BAD_SYNTAX
}RvSipOtherHeaderStringName;
```

RvSipPartyHeaderStringName

DESCRIPTION

Defines all Party header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipPartyHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_PARTY_DISPLAY_NAME,
    RVSIP_PARTY_TAG,
    RVSIP_PARTY_OTHER_PARAMS,
    RVSIP_PARTY_BAD_SYNTAX
}RvSipPartyHeaderStringName;
```

RvSipRAckHeaderStringName

DEFINITION

Defines all RAck header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipRAckHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_RACK_BAD_SYNTAX
}RvSipRAckHeaderStringName;
```

RvSipReferredByHeaderStringName

DESCRIPTION

Defines all the Referred-By header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipReferredByHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_REFERRED_BY_DISP_NAME,
    RVSIP_REFERRED_BY_OTHER_PARAMS,
    RVSIP_REFERRED_BY_CID_PARAM,
    RVSIP_REFERRED_BY_BAD_SYNTAX
}RvSipReferredByHeaderStringName;
```

RvSipReferToHeaderStringName

DESCRIPTION

Defines all Refer-To header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipReferToHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_REFER_TO_DISPLAY_NAME,
    RVSIP_REFER_TO_OTHER_PARAMS,
    RVSIP_REFER_TO_BAD_SYNTAX
}RvSipReferToHeaderStringName;
```

RvSipReplacesHeaderStringName

DESCRIPTION

Defines all Replaces header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipReplacesHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_REPLACES_CALL_ID,
    RVSIP_REPLACES_FROM_TAG,
    RVSIP_REPLACES_TO_TAG,
    RVSIP_REPLACES_OTHER_PARAMS,
    RVSIP_REPLACES_BAD_SYNTAX
}RvSipReplacesHeaderStringName;
```

RvSipRetryAfterHeaderStringName

DESCRIPTION

Defines all Retry-After header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipRetryAfterHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_RETRY_AFTER_COMMENT,
    RVSIP_RETRY_AFTER_PARAMS,
    RVSIP_RETRY_AFTER_BAD_SYNTAX
}RvSipRetryAfterHeaderStringName;
```

RvSipRouteHopHeaderStringName

DESCRIPTION

Defines all the Route-Hop header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipRouteHopHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_ROUTE_HOP_DISPLAY_NAME,
    RVSIP_ROUTE_HOP_OTHER_PARAMS,
    RVSIP_ROUTE_HOP_BAD_SYNTAX
}RvSipRouteHopHeaderStringName;
```

RvSipRSeqHeaderStringName

DESCRIPTION

Defines all RSeq header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipRSeqHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_RSEQ_BAD_SYNTAX
}RvSipRSeqHeaderStringName;
```

RvSipSessionExpiresHeaderStringName

DESCRIPTION

Defines all Session-Expires header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipSessionExpiresHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_SESSION_EXPIRES_OTHER_PARAMS,
    RVSIP_SESSION_EXPIRES_BAD_SYNTAX
}RvSipSessionExpiresHeaderStringName;
```

RvSipSubscriptionStateHeaderStringName

DESCRIPTION

Defines all Subscription-State header parameters that are kept in the object in string format. This enumeration is defined in the *RvSipSubscriptionStateHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_SUBSCRIPTION_STATE_SUBSTATE_VAL,
    RVSIP_SUBSCRIPTION_STATE_REASON,
    RVSIP_SUBSCRIPTION_STATE_OTHER_PARAMS,
    RVSIP_SUBSCRIPTION_STATE_BAD_SYNTAX
}RvSipSubscriptionStateHeaderStringName;
```

RvSipViaHeaderStringName

DESCRIPTION

Defines all Via header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipViaHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_VIA_TRANSPORT,
    RVSIP_VIA_HOST,
    RVSIP_VIA_MADDR_PARAM,
    RVSIP_VIA_RECEIVED_PARAM,
    RVSIP_VIA_BRANCH_PARAM,
    RVSIP_VIA_COMP_PARAM,
    RVSIP_VIA_SIGCOMPID_PARAM,
    RVSIP_VIA_OTHER_PARAMS,
    RVSIP_VIA_BAD_SYNTAX
}RvSipViaHeaderStringName;
```

RvSipReplacesHeaderStringName

DESCRIPTION

Defines all Replaces header parameters that are kept in the object in a string format. This enumeration is defined in the *RvSipReplacesHeader.h* header file.

SYNTAX

```
typedef enum{
    RVSIP_REPLACES_CALL_ID,
    RVSIP_REPLACES_FROM_TAG,
    RVSIP_REPLACES_TO_TAG,
    RVSIP_REPLACES_OTHER_PARAMS,
    RVSIP_REPLACES_BAD_SYNTAX
}RvSipReplacesHeaderStringName;
```

RvSipAuthenticationInfoHeaderStringName

DESCRIPTION

Defines all Authentication-Info header fields that are kept in the object in a string format. This enumeration is defined in the *RvSipAuthenticationInfoHeader.h* header file.

SYNTAX

```
RvStatus RvSipAuthenticationInfoHeaderConstructInMsg(
  IN  RvSipMsgHandle                      hSipMsg,
  IN  RvBool                              pushHeaderAtHead,
  OUT RvSipAuthenticationInfoHeaderHandle  *hHeader);
```

MESSAGE API TYPE DEFINITIONS

The Message API type definitions are:

- RvSipAddressType
- RvSipAuthAlgorithm
- RvSipAuthQopOption
- RvSipAuthScheme
- RvSipAuthStale
- RvSipAuthenticationHeaderType
- RvSipAuthIntegrityProtected
- RvSipAuthorizationHeaderType
- RvSipCompType
- RvSipContactAction
- RvSipDateMonth
- RvSipDateWeekDay
- RvSipDispositionType
- RvSipDispositionHandling
- RvSipExpiresFormat
- RvSipHeadersLocation
- RvSipHeaderType
- RvSipMediaType
- RvSipMediaSubType
- RvSipMethodType
- RvSipMsgHeadersOption
- RvSipMsgType
- RvSipReplacesEarlyFlagType
- RvSipRouteHopHeaderType
- RvSipSessionExpiresRefresherType
- RvSipSubscriptionReason
- RvSipSubscriptionSubstate
- RvSipTelUriEnumdiType
- RvSipTransport
- RvSipUserParam
- RvSipUrlScheme
- RvSipUrlLrType

RvSipAddressType

DESCRIPTION

The enumeration of the SIP address types for use in the address object.

SYNTAX

```
typedef enum{
    RVSIP_ADDRTYPE_UNDEFINED = -1,
    RVSIP_ADDRTYPE_ABS_URI,
    RVSIP_ADDRTYPE_URL,
    RVSIP_ADDRTYPE_TEL,
    RVSIP_ADDRTYPE_PRES,
    RVSIP_ADDRTYPE_IM,
    RVSIP_ADDRTYPE_DIAMETER
}RvSipAddressType;
```

RvSipAuthAlgorithm

DESCRIPTION

The enumeration used to indicate for the authentication algorithm type.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_ALGORITHM_UNDEFINED = -1,
    RVSIP_AUTH_ALGORITHM_MD5,
    RVSIP_AUTH_ALGORITHM_OTHER,
}RvSipAuthAlgorithm;
```

RvSipAuthQopOption

DESCRIPTION

The enumeration used to indicate the authentication Qop options.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_QOP_UNDEFINED = -1,
    RVSIP_AUTH_QOP_OTHER,
    RVSIP_AUTH_QOP_AUTH_ONLY,
    RVSIP_AUTH_QOP_AUTHINT_ONLY,
    RVSIP_AUTH_QOP_AUTH_AND_AUTHINT,
}RvSipAuthQopOption;
```

RvSipAuthScheme

DESCRIPTION

The enumeration used to indicate the authentication scheme.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_SCHEME_UNDEFINED = -1,
    RVSIP_AUTH_SCHEME_DIGEST,
    RVSIP_AUTH_SCHEME_OTHER,
}RvSipAuthScheme;
```

RvSipAuthStale

DESCRIPTION

The enumeration used to indicate the authentication stale parameter.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_STALE_UNDEFINED = -1,
    RVSIP_AUTH_STALE_FALSE,
    RVSIP_AUTH_STALE_TRUE,
}RvSipAuthStale;
```

RvSipAuthenticationHeaderType

DESCRIPTION

Indicates whether the Authentication header is a WWW–Authenticate header or a Proxy–Authenticate header. This enumeration is defined in the *RvSipAuthenticationHeader.h* file.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_UNDEFINED_AUTHENTICATION_HEADER = -1,
    RVSIP_AUTH_WWW_AUTHENTICATION_HEADER,
    RVSIP_AUTH_PROXY_AUTHENTICATION_HEADER
}RvSipAuthenticationHeaderType;
```

RvSipAuthIntegrityProtected

DESCRIPTION

This enumeration is for the authentication integrity-protected parameter

SYNTAX

```
typedef enum {
    RVSIP_AUTH_INTEGRITY_PROTECTED_UNDEFINED = -1,
    RVSIP_AUTH_INTEGRITY_PROTECTED_NO,
    RVSIP_AUTH_INTEGRITY_PROTECTED_YES
}RvSipAuthIntegrityProtected;
```

RvSipAuthorizationHeaderType

DESCRIPTION

Indicates whether the Authorization header is a WWW–Authorization header or a Proxy–Authorization header. This enumeration is defined in the *RvSipAuthorizationHeader.h* file.

SYNTAX

```
typedef enum{
    RVSIP_AUTH_UNDEFINED_AUTHORIZATION_HEADER = -1,
    RVSIP_AUTH_AUTHORIZATION_HEADER,
    RVSIP_AUTH_PROXY_AUTHORIZATION_HEADER
}RvSipAuthorizationHeaderType;
```

RvSipCompType

DESCRIPTION

Indicates the compression type of SIP messages.

SYNTAX

```
typedef enum{
    RVSIP_COMP_UNDEFINED = -1,
    RVSIP_COMP_SIGCOMP,
    RVSIP_COMP_OTHER
}RvSipCompType;
```

PARAMETERS

RVSIP_COMP_UNDEFINED

Undefined compression type.

RVSIP_COMP_SIGCOMP

SigComp compression.

RVSIP_COMP_OTHER

None of the above.

RvSipContactAction

DESCRIPTION

This enumeration is used to indicate the contact action.

SYNTAX

```
typedef enum{
    RVSIP_CONTACT_ACTION_UNDEFINED = -1,
    RVSIP_CONTACT_ACTION_PROXY,
    RVSIP_CONTACT_ACTION_REDIRECT
}RvSipContactAction;
```

RvSipDateMonth

DESCRIPTION

The enumeration used to indicate the month within a given date.

SYNTAX

```
typedef enum{
    RVSIP_MONTH_UNDEFINED = -1,
    RVSIP_MONTH_JAN,
    RVSIP_MONTH_FEB,
    RVSIP_MONTH_MAR,
    RVSIP_MONTH_APR,
    RVSIP_MONTH_MAY,
    RVSIP_MONTH_JUN,
    RVSIP_MONTH_JUL,
    RVSIP_MONTH_AUG,
    RVSIP_MONTH_SEP,
    RVSIP_MONTH_OCT,
    RVSIP_MONTH_NOV,
    RVSIP_MONTH_DEC
}RvSipDateMonth;
```

RvSipDateWeekDay

DESCRIPTION

The enumeration used to indicate the week day within a given date.

SYNTAX

```
typedef enum{
    RVSIP_WEEKDAY_UNDEFINED = -1,
    RVSIP_WEEKDAY_SUN,
    RVSIP_WEEKDAY_MON,
    RVSIP_WEEKDAY_TUE,
    RVSIP_WEEKDAY_WED,
    RVSIP_WEEKDAY_THU,
    RVSIP_WEEKDAY_FRI,
    RVSIP_WEEKDAY_SAT
}RvSipDateWeekDay;
```

RvSipDispositionType

DESCRIPTION

Indicates the disposition type of a Content-Disposition header.

SYNTAX

```
typedef enum{
    RVSIP_DISPOSITIONTYPE_UNDEFINED = -1,
    RVSIP_DISPOSITIONTYPE_RENDER,
    RVSIP_DISPOSITIONTYPE_SESSION,
    RVSIP_DISPOSITIONTYPE_ICON,
    RVSIP_DISPOSITIONTYPE_ALERT,
    RVSIP_DISPOSITIONTYPE_SIGNAL,
    RVSIP_DISPOSITIONTYPE_OTHER
}RvSipDispositionType;
```

RvSipDispositionHandling

DESCRIPTION

Indicates the handling type of a Content-Disposition header.

SYNTAX

```
typedef enum{
    RVSIP_DISPOSITIONHANDLING_UNDEFINED = -1,
    RVSIP_DISPOSITIONHANDLING_OPTIONAL,
    RVSIP_DISPOSITIONHANDLING_REQUIRED,
    RVSIP_DISPOSITIONHANDLING_OTHER
}RvSipDispositionHandling;
```

RvSipExpiresFormat

DESCRIPTION

The enumeration used to indicate the format of a given Expires header. If the header is not initialized the format is undefined. An initialized Expires header can either contain a Date header or an integer indicating delta-seconds time.

SYNTAX

```
typedef enum{
    RVSIP_EXPIRES_FORMAT_UNDEFINED = -1,
    RVSIP_EXPIRES_FORMAT_DELTA_SECONDS,
    RVSIP_EXPIRES_FORMAT_DATE
}RvSipExpiresFormat
```

RvSipHeadersLocation

DESCRIPTION

The enumeration used for retrieving parameters from the header list. The parameter indicate the location the header you wish to retrieve.

SYNTAX

```
typedef enum{
    RVSIP_FIRST_HEADER = 0,
    RVSIP_LAST_HEADER,
    RVSIP_NEXT_HEADER,
    RVSIP_PREV_HEADER
}RvSipHeadersLocation;
```

RvSipHeaderType

DESCRIPTION

Defines the header types of a SIP *message*.

SYNTAX

```
typedef enum{
    RVSIP_HEADERTYPE_UNDEFINED = -1,
    RVSIP_HEADERTYPE_ALLOW,
    RVSIP_HEADERTYPE_CONTACT,
    RVSIP_HEADERTYPE_VIA,
    RVSIP_HEADERTYPE_EXPIRES,
    RVSIP_HEADERTYPE_DATE,
    RVSIP_HEADERTYPE_ROUTE_HOP, /* 5*/
    RVSIP_HEADERTYPE_AUTHENTICATION,
    RVSIP_HEADERTYPE_AUTHORIZATION,
    RVSIP_HEADERTYPE_REFER_TO,
    RVSIP_HEADERTYPE_REFERRED_BY,
    RVSIP_HEADERTYPE_CONTENT_DISPOSITION, /*10 */
    RVSIP_HEADERTYPE_RETRY_AFTER,
    RVSIP_HEADERTYPE_OTHER,
    RVSIP_HEADERTYPE_RSEQ,
    RVSIP_HEADERTYPE_RACK,
    RVSIP_HEADERTYPE_TO, /* 15 */
    RVSIP_HEADERTYPE_FROM,
    RVSIP_HEADERTYPE_CSEQ,
    RVSIP_HEADERTYPE_SUBSCRIPTION_STATE,
    RVSIP_HEADERTYPE_EVENT,
    RVSIP_HEADERTYPE_ALLOW_EVENTS, /* 20 */
    RVSIP_HEADERTYPE_SESSION_EXPIRES,
    RVSIP_HEADERTYPE_MINSE,
    RVSIP_HEADERTYPE_REPLACES,
    RVSIP_HEADERTYPE_CONTENT_TYPE,
    RVSIP_HEADERTYPE_AUTHENTICATION_INFO,
```

```
    RVSIP_HEADERTYPE_REASON,
    RVSIP_HEADERTYPE_WARNING,
    RVSIP_HEADERTYPE_TIMESTAMP,
    RVSIP_HEADERTYPE_INFO,
    RVSIP_HEADERTYPE_ACCEPT, /* 30 */
    RVSIP_HEADERTYPE_ACCEPT_ENCODING,
    RVSIP_HEADERTYPE_ACCEPT_LANGUAGE,
    RVSIP_HEADERTYPE_REPLY_TO
    RVSIP_HEADERTYPE_P_PROFILE_KEY,
    RVSIP_HEADERTYPE_P_USER_DATABASE
}RvSipHeaderType;
```

REMARKS

Defines the header types that are stored in the headers list in a SIP message. The following headers are stored in the SIP message as a value and are not part of the headers list:

- To
- From
- CallId
- CSeq
- Content-Length
- Content-Type

Thus, the Content-Length, which is also a header type, is not in the enumeration.

RvSipMediaType

DESCRIPTION

Indicates the media type of the message body. The media type enumeration is found in the Content-Type header (The Content-Type has the format of Content-Type: media type/media sub type). When the media type is not set, it has the value of RVSIP_MEDIATYPE_UNDEFINED.

SYNTAX

```
typedef enum{
    RVSIP_MEDIATYPE_UNDEFINED = -1,
    RVSIP_MEDIATYPE_TEXT,
    RVSIP_MEDIATYPE_IMAGE,
    RVSIP_MEDIATYPE_AUDIO,
    RVSIP_MEDIATYPE_VIDEO,
    RVSIP_MEDIATYPE_APPLICATION,
    RVSIP_MEDIATYPE_MULTIPART,
    RVSIP_MEDIATYPE_MESSAGE,
    RVSIP_MEDIATYPE_OTHER
}RvSipMediaType;
```

RvSipMediaSubType

DESCRIPTION

Indicates the media sub-type of the message body. The media sub-type enumeration is found in the Content-Type header (The Content-Type has the format of Content-Type: media type/media sub type). When the media sub-type is not set, it has the value of RVSIP_MEDIASUBTYPE_UNDEFINED.

SYNTAX

```
typedef enum{
    RVSIP_MEDIASUBTYPE_UNDEFINED = -1,
    RVSIP_MEDIASUBTYPE_PLAIN,
    RVSIP_MEDIASUBTYPE_SDP,
    RVSIP_MEDIASUBTYPE_ISUP,
    RVSIP_MEDIASUBTYPE_QSIG,
    RVSIP_MEDIASUBTYPE_MIXED,
    RVSIP_MEDIASUBTYPE_ALTERNATIVE,
    RVSIP_MEDIASUBTYPE_DIGEST,
    RVSIP_MEDIASUBTYPE_RFC822,
    RVSIP_MEDIASUBTYPE_3GPP_IMS_XML,
    RVSIP_MEDIASUBTYPE_PIDF_XML,
    RVSIP_MEDIASUBTYPE_PARTIAL_PIDF_XML,
    RVSIP_MEDIASUBTYPE_WATCHERINFO_XML,
    RVSIP_MEDIASUBTYPE_RELATED,
    RVSIP_MEDIASUBTYPE_CSTA_XML,
    RVSIP_MEDIASUBTYPE_OTHER,
}RvSipMediaSubType;
```

RvSipMethodType

DESCRIPTION

Defines the methods supported by the SIP Stack.

SYNTAX

```
typedef enum{
    RVSIP_METHOD_UNDEFINED = -1,
    RVSIP_METHOD_INVITE,
    RVSIP_METHOD_ACK,
    RVSIP_METHOD_BYE,
    RVSIP_METHOD_REGISTER,
    RVSIP_METHOD_REFER,
    RVSIP_METHOD_NOTIFY,
    RVSIP_METHOD_OTHER,
    RVSIP_METHOD_PRACK,
    RVSIP_METHOD_CANCEL,
    RVSIP_METHOD_SUBSCRIBE
}RvSipMethodType;
```

RvSipMsgHeadersOption

DESCRIPTION

Defines which type of headers the user wishes to get in the following functions:

- RvSipMsgGetHeaderExt
- RvSipMsgGetHeaderByTypeExt
- RvSipMsgGetHeaderByNameExt

SYNTAX

```
typedef enum{
    RVSIP_MSG_HEADERS_OPTION_UNDEFINED = -1,
    RVSIP_MSG_HEADERS_OPTION_LEGAL_SYNTAX,
    RVSIP_MSG_HEADERS_OPTION_ALL
}RvSipMsgHeadersOption;
```

PARAMETERS

RVSIP_MSG_HEADERS_OPTION_LEGAL_SYNTAX

Only legal headers, with no syntax errors.

RVSIP_MSG_HEADERS_OPTION_ALL

All headers, bad-syntax or not.

RvSipMsgType

DESCRIPTION

Determines the type of message: request or response.

SYNTAX

```
typedef enum{
    RVSIP_MSG_UNDEFINED = -1,
    RVSIP_MSG_REQUEST,
    RVSIP_MSG_RESPONSE
}RvSipMsgType;
```

PARAMETERS

RVSIP_MSG_UNDEFINED

Undefined message type.

RVSIP_MSG_REQUEST

Request message type.

RVSIP_MSG_RESPONSE

Response message type.

RvSipReplacesEarlyFlagType

DESCRIPTION

Defines how the early-only parameter was received (early-only, early-only=true, early-only=1).

SYNTAX

```
typedef enum{
    RVSIP_REPLACES_EARLY_FLAG_TYPE_UNDEFINED = -1,
    RVSIP_REPLACES_EARLY_FLAG_TYPE_EARLY_ONLY_EMPTY,
    RVSIP_REPLACES_EARLY_FLAG_TYPE_EARLY_ONLY_1,
    RVSIP_REPLACES_EARLY_FLAG_TYPE_EARLY_ONLY_TRUE
}RvSipReplacesEarlyFlagType;
```

RvSipRouteHopHeaderType

DESCRIPTION

Defines whether the Route-Hop header represents a Route or Record-Route header. This enumeration is defined in the *RvSipRouteHopHeader.h* file.

SYNTAX

```
typedef enum{
    RVSIP_ROUTE_HOP_UNDEFINED_HEADER = -1,
    RVSIP_ROUTE_HOP_ROUTE_HEADER,
    RVSIP_ROUTE_HOP_RECORD_ROUTE_HEADER,
    RVSIP_ROUTE_HOP_PATH_HEADER,
    RVSIP_ROUTE_HOP_SERVICE_ROUTE_HEADER
}RvSipRouteHopHeaderType;
```

RvSipSessionExpiresRefresherType

DESCRIPTION

For use when the session timer feature is enabled.

SYNTAX

```
typedef enum{
    RVSIP_SESSION_EXPIRES_REFRESHER_NONE = -1,
    RVSIP_SESSION_EXPIRES_REFRESHER_UAC,
    RVSIP_SESSION_EXPIRES_REFRESHER_UAS
}RvSipSessionExpiresRefresherType;
```

PARAMETERS

RVSIP_SESSION_EXPIRES_REFRESHER_NONE

The refresher side of the call has not been set.

RVSIP_SESSION_EXPIRES_REFRESHER_UAC

The refresher of the call is the UAC.

RVSIP_SESSION_EXPIRES_REFRESHER_UAS

The refresher of the call is the UAS.

RvSipSubscriptionReason

DESCRIPTION

Indicates the reason in the Subscription-State header.

SYNTAX

```
typedef enum{
    RVSIP_SUBSCRIPTION_REASON_UNDEFINED = -1,
    RVSIP_SUBSCRIPTION_REASON_DEACTIVATED,
    RVSIP_SUBSCRIPTION_REASON_PROBATION,
    RVSIP_SUBSCRIPTION_REASON_REJECTED,
    RVSIP_SUBSCRIPTION_REASON_TIMEOUT,
    RVSIP_SUBSCRIPTION_REASON_GIVEUP,
    RVSIP_SUBSCRIPTION_REASON_NORESOURCE,
    RVSIP_SUBSCRIPTION_REASON_OTHER
}RvSipSubscriptionReason;
```

RvSipSubscriptionSubstate

DESCRIPTION

Indicates the substate in the Subscription-State header.

SYNTAX

```
typedef enum{
    RVSIP_SUBSCRIPTION_SUBSTATE_UNDEFINED = -1,
    RVSIP_SUBSCRIPTION_SUBSTATE_ACTIVE,
    RVSIP_SUBSCRIPTION_SUBSTATE_PENDING,
    RVSIP_SUBSCRIPTION_SUBSTATE_TERMINATED,
    RVSIP_SUBSCRIPTION_SUBSTATE_OTHER
}RvSipSubscriptionSubstate;
```

RvSipTelUriEnumdiType

DESCRIPTION

Indicates the enumdi type of a TEL URI address.

SYNTAX

```
typedef enum{
    RVSIP_ENUMDI_TYPE_UNDEFINED = -1,
    RVSIP_ENUMDI_TYPE_EXISTS_EMPTY
}RvSipTelUriEnumdiType;
```

PARAMETERS

RVSIP_ENUMDI_TYPE_UNDEFINED

The enumdi parameter was not found in the TEL URI address.

RVSIP_ENUMDI_TYPE_EXISTS_EMPTY

The enumdi parameter (";enumdi") was found in the TEL URI address.

RvSipTransport

DESCRIPTION

Determines the transport protocol to be used for sending SIP requests and responses.

SYNTAX

```
typedef enum{
    RVSIP_TRANSPORT_UNDEFINED = -1,
    RVSIP_TRANSPORT_UDP,
    RVSIP_TRANSPORT_TCP,
    RVSIP_TRANSPORT_SCTP,
    RVSIP_TRANSPORT_TLS,
    RVSIP_TRANSPORT_OTHER
}RvSipTransport;
```

PARAMETERS

RVSIP_TRANSPORT_UNDEFINED

Undefined transport protocol.

RVSIP_TRANSPORT_UDP

UDP transport protocol.

RVSIP_TRANSPORT_TCP

TCP transport protocol.

RVSIP_TRANSPORT_SCTP

SCTP transport protocol.

RVSIP_TRANSPORT_TLS

TLS transport protocol.

RVSIP_TRANSPORT_OTHER

Not one of the above transport protocols.

RvSipUserParam

DESCRIPTION

The enumeration of the user-param parameter of the SIP-URL.

SYNTAX

```
typedef enum{
    RVSIP_USERPARAM_UNDEFINED,
    RVSIP_USERPARAM_PHONE,
    RVSIP_USERPARAM_IP,
    RVSIP_USERPARAM_OTHER
}RvSipUserParam;
```

PARAMETERS

RVSIP_USERPARAM_UNDEFINED

Undefined user parameter.

RVSIP_USERPARAM_PHONE

Indicates that the SIP URL contains a phone number.

RVSIP_USERPARAM_IP

Indicates that the SIP URL contains either a host name or an IP address.

RVSIP_USERPARAM_OTHER

Not one of the above.

RvSipUrlScheme

DESCRIPTION

The enumeration of the SIP-URL scheme.

SYNTAX

```
typedef enum{
    RVSIP_URL_SCHEME_UNDEFINED = -1,
    RVSIP_URL_SCHEME_SIP,
    RVSIP_URL_SCHEME_SIPS
}RvSipUrlScheme;
```

PARAMETERS

RVSIP_URL_SCHEME_UNDEFINED

The URL scheme is undefined.

RVSIP_URL_SCHEME_SIP

The URL scheme is “SIP”.

RVSIP_URL_SCHEME_SIPS

The URL scheme is “SIPS”.

RvSipUrlLrType

DEFINITION

The enumeration that defines how the lr parameter is kept in the SIP address: lr, lr=1, or lr=true.

SYNTAX

```
typedef enum{
    RVSIP_URL_LR_TYPE_UNDEFINED = -1,
    RVSIP_URL_LR_TYPE_EMPTY,
    RVSIP_URL_LR_TYPE_1,
    RVSIP_URL_LR_TYPE_TRUE,
    RVSIP_URL_LR_TYPE_ON
}RvSipUrlLrType;
```

PARAMETERS

RVSIP_URL_LR_TYPE_UNDEFINED

The URL lr parameter type is undefined.

RVSIP_URL_LR_TYPE_EMPTY

The URL lr parameter is kept as “lr;”.

RVSIP_URL_LR_TYPE_1

The URL lr parameter is kept as “lr=1;”.

RVSIP_URL_LR_TYPE_TRUE

The URL lr parameter is kept as “lr=true”.

RVSIP_URL_LR_TYPE_ON

The URL lr parameter is kept as “lr=on;”.

INDEX

R

RvSipAcceptEncodingHeaderHandle 909
RvSipAcceptHeaderHandle 908
RvSipAcceptLanguageHeaderHandle 910
RvSipAddrAbsUriGetIdentifier() 142
RvSipAddrAbsUriGetScheme() 140
RvSipAddrAbsUriSetIdentifier() 143
RvSipAddrAbsUriSetScheme() 141
RvSipAddrConstruct() 78
RvSipAddrConstructAndParse() 84
RvSipAddrConstructInAuthorizationHeader() 74
RvSipAddrConstructInContactHeader() 73
RvSipAddrConstructInContentIDHeader() 83
RvSipAddrConstructInContentTypeHeader() 82
RvSipAddrConstructInPartyHeader() 72
RvSipAddrConstructInPProfileKeyHeader() 86
RvSipAddrConstructInPUserDatabaseHeader() 87
RvSipAddrConstructInReferredByHeader() 77
RvSipAddrConstructInReferToHeader() 76
RvSipAddrConstructInRouteHopHeader() 75
RvSipAddrConstructInStartLine() 71
RvSipAddrCopy() 79
RvSipAddrDiameterUriGetHost() 165
RvSipAddrDiameterUriGetOtherParams() 173
RvSipAddrDiameterUriGetPortNum() 167
RvSipAddrDiameterUriGetProtocolType() 164
RvSipAddrDiameterUriGetStrTransport() 170
RvSipAddrDiameterUriGetTransport() 169
RvSipAddrDiameterUriIsSecure() 88
RvSipAddrDiameterUriSetHost() 166
RvSipAddrDiameterUriSetIsSecure() 162
RvSipAddrDiameterUriSetOtherParams() 172
RvSipAddrDiameterUriSetPortNum() 168
RvSipAddrDiameterUriSetProtocolType() 163
RvSipAddrDiameterUriSetTransport() 171
RvSipAddrEncode() 80
RvSipAddressHandle 911
RvSipAddressStringName 949
RvSipAddressType 982
RvSipAddrGetAddressTypeFromString() 94
RvSipAddrGetAddrType() 91
RvSipAddrGetRpoolString() 92
RvSipAddrGetStringLength() 90
RvSipAddrIsEqual() 96
RvSipAddrParse() 81
RvSipAddrSetRpoolString() 93
RvSipAddrTelUriGetContext() 156
RvSipAddrTelUriGetEnumdiParamType() 158
RvSipAddrTelUriGetExtension() 150
RvSipAddrTelUriGetIsdnSubAddr() 152
RvSipAddrTelUriGetIsGlobalPhoneNumber() 146
RvSipAddrTelUriGetOtherParams() 160
RvSipAddrTelUriGetPhoneNumber() 148
RvSipAddrTelUriGetPostDial() 154
RvSipAddrTelUriSetContext() 155

RvSipAddrTelUriSetEnumdiParamType() 157
RvSipAddrTelUriSetExtension() 149
RvSipAddrTelUriSetIsdnSubAddr() 151
RvSipAddrTelUriSetIsGlobalPhoneNumber() 145
RvSipAddrTelUriSetOtherParams() 159
RvSipAddrTelUriSetPhoneNumber() 147
RvSipAddrTelUriSetPostDial() 153
RvSipAddrUrlGetCompParam() 126
RvSipAddrUrlGetCPC() 133
RvSipAddrUrlGetGr() 137
RvSipAddrUrlGetHeaders() 122
RvSipAddrUrlGetHeadersList() 124
RvSipAddrUrlGetHost() 101
RvSipAddrUrlGetLrParam() 107
RvSipAddrUrlGetLrParamType() 109
RvSipAddrUrlGetMaddrParam() 117
RvSipAddrUrlGetMethod() 111
RvSipAddrUrlGetOtherParams() 129
RvSipAddrUrlGetPortNum() 103
RvSipAddrUrlGetStrCompParam() 127
RvSipAddrUrlGetStrCPC() 134
RvSipAddrUrlGetStrGr() 136
RvSipAddrUrlGetStrMethod() 112
RvSipAddrUrlGetStrTransport() 115
RvSipAddrUrlGetStrUserParam() 120
RvSipAddrUrlGetTransport() 114
RvSipAddrUrlGetTtlNum() 105
RvSipAddrUrlGetUser() 99
RvSipAddrUrlGetUserParam() 119
RvSipAddrUrlIsSecure() 131
RvSipAddrUrlSetCompParam() 128
RvSipAddrUrlSetCPC() 135
RvSipAddrUrlSetGr() 138
RvSipAddrUrlSetHeaders() 123
RvSipAddrUrlSetHeadersList() 125
RvSipAddrUrlSetHost() 102
RvSipAddrUrlSetIsSecure() 132
RvSipAddrUrlSetLrParam() 108
RvSipAddrUrlSetLrParamType() 110
RvSipAddrUrlSetMaddrParam() 118
RvSipAddrUrlSetMethod() 113
RvSipAddrUrlSetOtherParams() 130
RvSipAddrUrlSetPortNum() 104
RvSipAddrUrlSetTransport() 116
RvSipAddrUrlSetTtlNum() 106
RvSipAddrUrlSetUser() 100
RvSipAddrUrlSetUserParam() 121
RvSipAllowEventsHeaderConstruct() 196
RvSipAllowEventsHeaderConstructInMsg() 195
RvSipAllowEventsHeaderCopy() 197
RvSipAllowEventsHeaderEncode() 198
RvSipAllowEventsHeaderFix() 201
RvSipAllowEventsHeaderGetCompactForm() 214
RvSipAllowEventsHeaderGetEventPackage() 206
RvSipAllowEventsHeaderGetEventTemplate() 208
RvSipAllowEventsHeaderGetRpoolString() 204
RvSipAllowEventsHeaderGetStrBadSyntax() 210
RvSipAllowEventsHeaderGetStringLength() 203
RvSipAllowEventsHeaderHandle 912
RvSipAllowEventsHeaderParse() 199
RvSipAllowEventsHeaderParseValue() 200
RvSipAllowEventsHeaderSetCompactForm() 213
RvSipAllowEventsHeaderSetEventPackage() 207
RvSipAllowEventsHeaderSetEventTemplate() 209
RvSipAllowEventsHeaderSetRpoolString() 205
RvSipAllowEventsHeaderSetStrBadSyntax() 212
RvSipAllowEventsHeaderStringName 951
RvSipAllowHeaderConstruct() 178
RvSipAllowHeaderConstructInMsg() 177
RvSipAllowHeaderCopy() 179
RvSipAllowHeaderEncode() 180
RvSipAllowHeaderFix() 183

RvSipAllowHeaderGetMethodType() 186
RvSipAllowHeaderGetStrBadSyntax() 189
RvSipAllowHeaderGetStringLength() 185
RvSipAllowHeaderGetStrMethodType() 187
RvSipAllowHeaderHandle 913
RvSipAllowHeaderParse() 181
RvSipAllowHeaderParseValue() 182
RvSipAllowHeaderSetMethodType() 188
RvSipAllowHeaderSetStrBadSyntax() 191
RvSipAllowHeaderStringName 952
RvSipAuthAlgorithm 983
RvSipAuthenticationHeaderConstruct() 218
RvSipAuthenticationHeaderConstructInMsg() 217
RvSipAuthenticationHeaderCopy() 219
RvSipAuthenticationHeaderEncode() 220
RvSipAuthenticationHeaderFix() 223
RvSipAuthenticationHeaderGetAKAVersion() 257
RvSipAuthenticationHeaderGetAuthAlgorithm() 242
RvSipAuthenticationHeaderGetAuthScheme() 227
RvSipAuthenticationHeaderGetDomain() 232
RvSipAuthenticationHeaderGetHeaderType() 247
RvSipAuthenticationHeaderGetNonce() 234
RvSipAuthenticationHeaderGetOpaque() 236
RvSipAuthenticationHeaderGetOtherParams() 249
RvSipAuthenticationHeaderGetQopOption() 245
RvSipAuthenticationHeaderGetRealm() 230
RvSipAuthenticationHeaderGetRpoolString() 252
RvSipAuthenticationHeaderGetStale() 240
RvSipAuthenticationHeaderGetStrAuthAlgorithm() 243
RvSipAuthenticationHeaderGetStrAuthScheme() 228
RvSipAuthenticationHeaderGetStrBadSyntax() 254
RvSipAuthenticationHeaderGetStrCipherKey() 261
RvSipAuthenticationHeaderGetStringLength() 226
RvSipAuthenticationHeaderGetStrIntegrityKey() 259
RvSipAuthenticationHeaderGetStrQop() 238
RvSipAuthenticationHeaderHandle 914
RvSipAuthenticationHeaderParse() 221
RvSipAuthenticationHeaderParseValue() 222
RvSipAuthenticationHeaderSetAKAVersion() 258
RvSipAuthenticationHeaderSetAuthAlgorithm() 244
RvSipAuthenticationHeaderSetAuthScheme() 229
RvSipAuthenticationHeaderSetDomain() 233
RvSipAuthenticationHeaderSetHeaderType() 248
RvSipAuthenticationHeaderSetNonce() 235
RvSipAuthenticationHeaderSetOpaque() 237
RvSipAuthenticationHeaderSetOtherParams() 251
RvSipAuthenticationHeaderSetQopOption() 246
RvSipAuthenticationHeaderSetRealm() 231
RvSipAuthenticationHeaderSetRpoolString() 253
RvSipAuthenticationHeaderSetStale() 241
RvSipAuthenticationHeaderSetStrBadSyntax() 256
RvSipAuthenticationHeaderSetStrCipherKey() 262
RvSipAuthenticationHeaderSetStrIntegrityKey() 260
RvSipAuthenticationHeaderSetStrQop() 239
RvSipAuthenticationHeaderStringName 953
RvSipAuthenticationHeaderType 987
RvSipAuthenticationInfoHeaderConstruct() 266
RvSipAuthenticationInfoHeaderConstructInMsg() 265
RvSipAuthenticationInfoHeaderCopy() 267

RvSipAuthenticationInfoHeaderEncode() 268
RvSipAuthenticationInfoHeaderFix() 271
RvSipAuthenticationInfoHeaderGetCNonce() 276
RvSipAuthenticationInfoHeaderGetNextNonce() 274
RvSipAuthenticationInfoHeaderGetNonceCount() 280
RvSipAuthenticationInfoHeaderGetQopOption() 282
RvSipAuthenticationInfoHeaderGetResponseAuth() 278
RvSipAuthenticationInfoHeaderGetRpoolString() 288
RvSipAuthenticationInfoHeaderGetStrBadSyntax() 285
RvSipAuthenticationInfoHeaderGetStringLength() 273
RvSipAuthenticationInfoHeaderGetStrQop() 283
RvSipAuthenticationInfoHeaderHandle 915
RvSipAuthenticationInfoHeaderParse() 269
RvSipAuthenticationInfoHeaderParseValue() 270
RvSipAuthenticationInfoHeaderSetCNonce() 277
RvSipAuthenticationInfoHeaderSetNextNonce() 275
RvSipAuthenticationInfoHeaderSetNonceCount() 281
RvSipAuthenticationInfoHeaderSetQopOption() 284
RvSipAuthenticationInfoHeaderSetResponseAuth() 279
RvSipAuthenticationInfoHeaderSetRpoolString() 289
RvSipAuthenticationInfoHeaderSetStrBadSyntax() 287
RvSipAuthenticationInfoHeaderStringName 980
RvSipAuthIntegrityProtected 988
RvSipAuthorizationHeaderConstruct() 294
RvSipAuthorizationHeaderConstructInMsg() 293
RvSipAuthorizationHeaderCopy() 295
RvSipAuthorizationHeaderEncode() 296
RvSipAuthorizationHeaderFix() 299
RvSipAuthorizationHeaderGetAuthAlgorithm() 318
RvSipAuthorizationHeaderGetAuthScheme() 305
RvSipAuthorizationHeaderGetCNonce() 325
RvSipAuthorizationHeaderGetCredentialIdentifier() 303
RvSipAuthorizationHeaderGetDigestUri() 327
RvSipAuthorizationHeaderGetHeaderType() 329
RvSipAuthorizationHeaderGetIntegrityProtected() 341
RvSipAuthorizationHeaderGetNonce() 310
RvSipAuthorizationHeaderGetNonceCount() 316
RvSipAuthorizationHeaderGetOpaque() 312
RvSipAuthorizationHeaderGetOtherParams() 331
RvSipAuthorizationHeaderGetQopOption() 321
RvSipAuthorizationHeaderGetRealm() 308
RvSipAuthorizationHeaderGetResponse() 323
RvSipAuthorizationHeaderGetRpoolString() 335
RvSipAuthorizationHeaderGetStrAuthAlgorithm() 319
RvSipAuthorizationHeaderGetStrAuthScheme() 306
RvSipAuthorizationHeaderGetStrAuts() 339
RvSipAuthorizationHeaderGetStrBadSyntax() 336
RvSipAuthorizationHeaderGetStringLength() 302
RvSipAuthorizationHeaderGetUserName() 314
RvSipAuthorizationHeaderHandle 916
RvSipAuthorizationHeaderParse() 297
RvSipAuthorizationHeaderParseValue() 298
RvSipAuthorizationHeaderSetAuthAlgorithm() 320
RvSipAuthorizationHeaderSetAuthScheme() 307

RvSipAuthorizationHeaderSetCNonce() 326
RvSipAuthorizationHeaderSetDigestUri() 328
RvSipAuthorizationHeaderSetHeaderType() 330
RvSipAuthorizationHeaderSetIntegrityProtected() 342
RvSipAuthorizationHeaderSetNonce() 311
RvSipAuthorizationHeaderSetNonceCount() 317
RvSipAuthorizationHeaderSetOpaque() 313
RvSipAuthorizationHeaderSetOtherParams() 333
RvSipAuthorizationHeaderSetQopOption() 322
RvSipAuthorizationHeaderSetRealm() 309
RvSipAuthorizationHeaderSetResponse() 324
RvSipAuthorizationHeaderSetRpoolString() 334
RvSipAuthorizationHeaderSetStrAuts() 340
RvSipAuthorizationHeaderSetStrBadSyntax() 338
RvSipAuthorizationHeaderSetUserName() 315
RvSipAuthorizationHeaderStringName 954
RvSipAuthorizationHeaderType 989
RvSipAuthQopOption 984
RvSipAuthScheme 985
RvSipAuthStale 986
RvSipBodyConstruct 831
RvSipBodyConstruct() 831
RvSipBodyConstructInBodyPart 830
RvSipBodyConstructInBodyPart() 830
RvSipBodyConstructInMsg 829
RvSipBodyConstructInMsg() 829
RvSipBodyCopy() 832
RvSipBodyEncode() 833
RvSipBodyGetBodyPart() 844
RvSipBodyGetBodyStr() 841
RvSipBodyGetBodyStrLength() 840
RvSipBodyGetContentType() 838
RvSipBodyGetRpoolString() 849
RvSipBodyHandle 917
RvSipBodyMultipartParse() 835
RvSipBodyPartHandle 918
RvSipBodyPushBodyPart() 846
RvSipBodyRemoveBodyPart() 848
RvSipBodySetBodyStr() 843
RvSipBodySetContentType() 839
RvSipBodySetRpoolString() 850
RvSipCompType 990
RvSipContactAction 991
RvSipContactHeaderConstruct() 346
RvSipContactHeaderConstructInMsg() 345
RvSipContactHeaderCopy() 347
RvSipContactHeaderEncode() 348
RvSipContactHeaderFix() 351
RvSipContactHeaderGetAction() 363
RvSipContactHeaderGetAddrSpec() 357
RvSipContactHeaderGetCompactForm() 375
RvSipContactHeaderGetDisplayName() 355
RvSipContactHeaderGetExpires() 359
RvSipContactHeaderGetOtherParams() 367
RvSipContactHeaderGetPubGruu() 380
RvSipContactHeaderGetQVal() 361
RvSipContactHeaderGetRegIDNum() 384
RvSipContactHeaderGetRpoolString() 369
RvSipContactHeaderGetStrBadSyntax() 371
RvSipContactHeaderGetStrFeatureTag() 376
RvSipContactHeaderGetStringLength() 354
RvSipContactHeaderGetTempGruu() 382
RvSipContactHeaderHandle 921
RvSipContactHeaderIsFeatureTagActive() 352
RvSipContactHeaderIsStar() 366
RvSipContactHeaderParse() 349
RvSipContactHeaderParseValue() 350
RvSipContactHeaderSetAction() 364
RvSipContactHeaderSetAddrSpec() 358
RvSipContactHeaderSetCompactForm() 374
RvSipContactHeaderSetDisplayName() 356
RvSipContactHeaderSetExpires() 360

RvSipContactHeaderSetFeatureTagActivation() 379
RvSipContactHeaderSetOtherParams() 368
RvSipContactHeaderSetPubGruu() 381
RvSipContactHeaderSetQVal() 362
RvSipContactHeaderSetRegIDNum() 385
RvSipContactHeaderSetRpoolString() 370
RvSipContactHeaderSetStar() 365
RvSipContactHeaderSetStrBadSyntax() 373
RvSipContactHeaderSetStrFeatureTag() 378
RvSipContactHeaderSetTempGruu() 383
RvSipContactHeaderStringName 955
RvSipContentDispositionHeaderConstruct() 391
RvSipContentDispositionHeaderConstructInBodyPart() 390
RvSipContentDispositionHeaderConstructInMsg() 389
RvSipContentDispositionHeaderCopy() 392
RvSipContentDispositionHeaderEncode() 393
RvSipContentDispositionHeaderFix() 396
RvSipContentDispositionHeaderGetHandling() 402
RvSipContentDispositionHeaderGetOtherParams() 405
RvSipContentDispositionHeaderGetStrBadSyntax() 408
RvSipContentDispositionHeaderGetStrHandling() 403
RvSipContentDispositionHeaderGetStringLength() 398
RvSipContentDispositionHeaderGetStrType() 400
RvSipContentDispositionHeaderGetType() 399
RvSipContentDispositionHeaderHandle 919
RvSipContentDispositionHeaderParse() 394
RvSipContentDispositionHeaderParseValue() 395
RvSipContentDispositionHeaderSetHandling() 404
RvSipContentDispositionHeaderSetOtherParams() 407
RvSipContentDispositionHeaderSetStrBadSyntax() 410
RvSipContentDispositionHeaderSetType() 401
RvSipContentDispositionHeaderStringName 957
RvSipContentIDHeaderConstruct() 454
RvSipContentIDHeaderConstructInBody() 453
RvSipContentIDHeaderCopy() 455
RvSipContentIDHeaderEncode() 456
RvSipContentIDHeaderFieldName 960
RvSipContentIDHeaderFix() 459
RvSipContentIDHeaderGetAddrSpec() 462
RvSipContentIDHeaderGetStrBadSyntax() 464
RvSipContentIDHeaderGetStringLength() 461
RvSipContentIDHeaderHandle 920
RvSipContentIDHeaderParse() 457
RvSipContentIDHeaderParseValue() 458
RvSipContentIDHeaderSetAddrSpec() 463
RvSipContentIDHeaderSetStrBadSyntax() 466
RvSipContentIDHeaderStringName 959
RvSipContentTypeHeaderConstruct() 414
RvSipContentTypeHeaderConstructInBody() 413
RvSipContentTypeHeaderCopy() 415
RvSipContentTypeHeaderEncode() 416
RvSipContentTypeHeaderFix() 419
RvSipContentTypeHeaderGetBase() 432
RvSipContentTypeHeaderGetBoundary() 428
RvSipContentTypeHeaderGetCompactForm() 441
RvSipContentTypeHeaderGetMediaSubType() 425
RvSipContentTypeHeaderGetMediaType() 422
RvSipContentTypeHeaderGetOtherParams() 434
RvSipContentTypeHeaderGetStart() 442
RvSipContentTypeHeaderGetStrBadSyntax() 437

RvSipContentTypeHeaderGetStringLength() 421
RvSipContentTypeHeaderGetStrMediaSubType() 426
RvSipContentTypeHeaderGetStrMediaType() 423
RvSipContentTypeHeaderGetVersion() 430
RvSipContentTypeHeaderHandle 922
RvSipContentTypeHeaderParse() 417
RvSipContentTypeHeaderParseValue() 418
RvSipContentTypeHeaderSetBase() 433
RvSipContentTypeHeaderSetBoundary() 429
RvSipContentTypeHeaderSetCompactForm() 440
RvSipContentTypeHeaderSetMediaSubType() 427
RvSipContentTypeHeaderSetMediaType() 424
RvSipContentTypeHeaderSetOtherParams() 436
RvSipContentTypeHeaderSetStart() 443
RvSipContentTypeHeaderSetStrBadSyntax() 439
RvSipContentTypeHeaderSetVersion() 431
RvSipContentTypeHeaderStringName 958
RvSipContentTypeHeaderTypeParamGetMediaSubType() 446
RvSipContentTypeHeaderTypeParamGetMediaType() 444
RvSipContentTypeHeaderTypeParamGetStrMediaSubType() 447
RvSipContentTypeHeaderTypeParamGetStrMediaType() 445
RvSipContentTypeHeaderTypeParamSetMediaSubType() 450
RvSipContentTypeHeaderTypeParamSetMediaType() 449
RvSipCSeqHeaderConstruct() 471
RvSipCSeqHeaderConstructInMsg() 470
RvSipCSeqHeaderConstructInRAckHeader() 472
RvSipCSeqHeaderCopy() 473
RvSipCSeqHeaderEncode() 474
RvSipCSeqHeaderFix() 477
RvSipCSeqHeaderGetMethodType() 482
RvSipCSeqHeaderGetStep() 480
RvSipCSeqHeaderGetStrBadSyntax() 485
RvSipCSeqHeaderGetStringLength() 479
RvSipCSeqHeaderGetStrMethodType() 483
RvSipCSeqHeaderHandle 923
RvSipCSeqHeaderParse() 475
RvSipCSeqHeaderParseValue() 476
RvSipCSeqHeaderSetMethodType() 484
RvSipCSeqHeaderSetStep() 481
RvSipCSeqHeaderSetStrBadSyntax() 487
RvSipCSeqHeaderStringName 961
RvSipDateConstructInExpiresHeader() 492
RvSipDateConstructInRetryAfterHeader() 493
RvSipDateHeaderConstruct() 494
RvSipDateHeaderConstructInMsg() 491
RvSipDateHeaderCopy() 495
RvSipDateHeaderEncode() 496
RvSipDateHeaderFix() 499
RvSipDateHeaderGetDay() 503
RvSipDateHeaderGetHour() 509
RvSipDateHeaderGetMinute() 511
RvSipDateHeaderGetMonth() 505
RvSipDateHeaderGetSecond() 513
RvSipDateHeaderGetStrBadSyntax() 516
RvSipDateHeaderGetStringLength() 515
RvSipDateHeaderGetWeekDay() 501
RvSipDateHeaderGetYear() 507
RvSipDateHeaderHandle 924
RvSipDateHeaderParse() 497
RvSipDateHeaderParseValue() 498
RvSipDateHeaderSetDay() 504
RvSipDateHeaderSetHour() 510
RvSipDateHeaderSetMinute() 512
RvSipDateHeaderSetMonth() 506
RvSipDateHeaderSetSecond() 514
RvSipDateHeaderSetStrBadSyntax() 518
RvSipDateHeaderSetWeekDay() 502
RvSipDateHeaderSetYear() 508
RvSipDateHeaderStringName 962

RvSipDateMonth 992
RvSipDateWeekDay 993
RvSipDispositionHandling 995
RvSipDispositionType 994
RvSipEventHeaderConstruct() 522
RvSipEventHeaderConstructInMsg() 521
RvSipEventHeaderCopy() 523
RvSipEventHeaderEncode() 524
RvSipEventHeaderFix() 527
RvSipEventHeaderGetCompactForm() 545
RvSipEventHeaderGetEventId() 539
RvSipEventHeaderGetEventPackage() 533
RvSipEventHeaderGetEventParam() 537
RvSipEventHeaderGetEventTemplate() 535
RvSipEventHeaderGetRpoolString() 531
RvSipEventHeaderGetStrBadSyntax() 541
RvSipEventHeaderGetStringLength() 530
RvSipEventHeaderHandle 925
RvSipEventHeaderIsEqual() 528
RvSipEventHeaderParse() 525, 528
RvSipEventHeaderParseValue() 526
RvSipEventHeaderSetCompactForm() 544
RvSipEventHeaderSetEventId() 540
RvSipEventHeaderSetEventPackage() 534
RvSipEventHeaderSetEventParam() 538
RvSipEventHeaderSetEventTemplate() 536
RvSipEventHeaderSetRpoolString() 532
RvSipEventHeaderSetStrBadSyntax() 543
RvSipEventHeaderStringName 963
RvSipExpiresConstructInContactHeader() 550
RvSipExpiresFormat 996
RvSipExpiresHeaderConstruct() 551
RvSipExpiresHeaderConstructInMsg() 549
RvSipExpiresHeaderCopy() 552
RvSipExpiresHeaderEncode() 553
RvSipExpiresHeaderFix() 556
RvSipExpiresHeaderGetDateHandle() 561
RvSipExpiresHeaderGetDeltaSeconds() 559
RvSipExpiresHeaderGetFormat() 558
RvSipExpiresHeaderGetStrBadSyntax() 564
RvSipExpiresHeaderGetStringLength() 563
RvSipExpiresHeaderHandle 926
RvSipExpiresHeaderParse() 554
RvSipExpiresHeaderParseValue() 555
RvSipExpiresHeaderSetDateHandle() 562
RvSipExpiresHeaderSetDeltaSeconds() 560
RvSipExpiresHeaderSetStrBadSyntax() 566
RvSipExpiresHeaderStringName 964
RvSipFromHeaderConstructInMsg() 608
RvSipHeaderEncode() 898
RvSipHeaderIsBadSyntax() 901
RvSipHeaderParse() 899
RvSipHeaderParseValue() 900
RvSipHeadersListElemHandle 927
RvSipHeadersListElemPosHandle 927
RvSipHeadersLocation 997
RvSipHeaderType 998
RvSipInfoHeaderHandle 928
RvSipMediaSubType 1001
RvSipMediaType 1000
RvSipMessageStringName 965
RvSipMethodType 1002
RvSipMinSEHeaderConstruct() 570
RvSipMinSEHeaderConstructInMsg() 569
RvSipMinSEHeaderCopy() 571
RvSipMinSEHeaderEncode() 572
RvSipMinSEHeaderFix() 575
RvSipMinSEHeaderGetDeltaSeconds() 577
RvSipMinSEHeaderGetOtherParams() 579
RvSipMinSEHeaderGetStrBadSyntax() 582
RvSipMinSEHeaderGetStringLength() 581
RvSipMinSEHeaderHandle 929
RvSipMinSEHeaderParse() 573
RvSipMinSEHeaderParseValue() 574
RvSipMinSEHeaderSetDeltaSeconds() 578
RvSipMinSEHeaderSetOtherParams() 580
RvSipMinSEHeaderSetStrBadSyntax() 584
RvSipMinSEHeaderStringName 966
RvSipMsgConstruct() 5
RvSipMsgConstructHeaderInMsgByType() 10

RvSipMsgCopy() 7
RvSipMsgDestruct() 6
RvSipMsgEncode() 8
RvSipMsgForceCompactForm() 11
RvSipMsgGetBadSyntaxHeaders() 37
RvSipMsgGetBody() 54
RvSipMsgGetBodyObject() 57
RvSipMsgGetCallIdCompactForm() 65
RvSipMsgGetCallIdHeader() 50
RvSipMsgGetContentLengthCompactForm() 67
RvSipMsgGetContentLengthHeader() 46
RvSipMsgGetContentTypeHeader() 52
RvSipMsgGetCSeqHeader() 48
RvSipMsgGetFromHeader() 44
RvSipMsgGetHeader() 25
RvSipMsgGetHeaderByName() 29
RvSipMsgGetHeaderByNameExt() 35
RvSipMsgGetHeaderByType() 27
RvSipMsgGetHeaderByTypeExt() 33
RvSipMsgGetHeaderExt() 31
RvSipMsgGetMsgType() 15
RvSipMsgGetReasonPhrase() 23
RvSipMsgGetRequestMethod() 16
RvSipMsgGetRequestUri() 19
RvSipMsgGetRpoolString() 59
RvSipMsgGetStatusCode() 21
RvSipMsgGetStrBadSyntaxStartLine() 61
RvSipMsgGetStringLength() 14
RvSipMsgGetStrRequestMethod() 17
RvSipMsgGetToHeader() 42
RvSipMsgHandle 930
RvSipMsgHeadersOption 1003
RvSipMsgMgrHandle 931
RvSipMsgParse() 9
RvSipMsgPushHeader() 39
RvSipMsgRemoveHeaderAt() 41
RvSipMsgSetBody() 56
RvSipMsgSetBodyObject() 58
RvSipMsgSetCallIdCompactForm() 64
RvSipMsgSetCallIdHeader() 51
RvSipMsgSetContentLengthCompactForm() 66
RvSipMsgSetContentLengthHeader() 47
RvSipMsgSetContentTypeHeader() 53
RvSipMsgSetCSeqHeader() 49
RvSipMsgSetFromHeader() 45
RvSipMsgSetMethodInRequestLine() 18
RvSipMsgSetReasonPhrase() 24
RvSipMsgSetRequestUri() 20
RvSipMsgSetRpoolString() 60
RvSipMsgSetStatusCode() 22
RvSipMsgSetToHeader() 43
RvSipMsgStartLineFix() 63
RvSipMsgType 1004
RvSipOtherHeaderConstruct() 589
RvSipOtherHeaderConstructInBodyPart() 588
RvSipOtherHeaderConstructInMsg() 587
RvSipOtherHeaderCopy() 590
RvSipOtherHeaderEncode() 591
RvSipOtherHeaderGetName() 595
RvSipOtherHeaderGetRpoolString() 599
RvSipOtherHeaderGetStrBadSyntax() 601
RvSipOtherHeaderGetStringLength() 594
RvSipOtherHeaderGetValue() 597
RvSipOtherHeaderHandle 932
RvSipOtherHeaderParse() 592
RvSipOtherHeaderSetName() 596
RvSipOtherHeaderSetRpoolString() 600
RvSipOtherHeaderSetStrBadSyntax() 603
RvSipOtherHeaderSetValue() 598
RvSipOtherHeaderStringName 967
RvSipPartyHeaderConstruct() 609
RvSipPartyHeaderCopy() 610
RvSipPartyHeaderEncode() 611
RvSipPartyHeaderFix() 616
RvSipPartyHeaderGetAddrSpec() 624
RvSipPartyHeaderGetCompactForm() 636
RvSipPartyHeaderGetDisplayName() 622
RvSipPartyHeaderGetOtherParams() 626
RvSipPartyHeaderGetRpoolString() 630
RvSipPartyHeaderGetStrBadSyntax() 632

RvSipPartyHeaderGetStringLength() 619
RvSipPartyHeaderGetTag() 620
RvSipPartyHeaderGetType() 628
RvSipPartyHeaderHandle 933
RvSipPartyHeaderParse() 613
RvSipPartyHeaderParseValue() 614
RvSipPartyHeaderSetAddrSpec() 625
RvSipPartyHeaderSetCompactForm() 635
RvSipPartyHeaderSetDisplayName() 623
RvSipPartyHeaderSetOtherParams() 627
RvSipPartyHeaderSetRpoolString() 631
RvSipPartyHeaderSetStrBadSyntax() 634
RvSipPartyHeaderSetTag() 621
RvSipPartyHeaderSetType() 629
RvSipPartyHeaderStringName 968
RvSipPartyIsEqual() 617
RvSipRAckHeaderConstruct() 640
RvSipRAckHeaderConstructInMsg() 639
RvSipRAckHeaderCopy() 641
RvSipRAckHeaderEncode() 642
RvSipRAckHeaderFix() 645
RvSipRAckHeaderGetCSeqHandle() 650
RvSipRAckHeaderGetResponseNum() 648
RvSipRAckHeaderGetStrBadSyntax() 653
RvSipRAckHeaderGetStringLength() 652
RvSipRAckHeaderHandle 934
RvSipRAckHeaderIsResponseNumInitialized( ) 646
RvSipRAckHeaderParse() 643
RvSipRAckHeaderParseValue() 644
RvSipRAckHeaderSetCSeqHandle() 651
RvSipRAckHeaderSetResponseNum() 649
RvSipRAckHeaderSetStrBadSyntax() 655
RvSipRAckHeaderStringName 969
RvSipReasonHeaderHandle 935
RvSipReferredByheaderConstruct() 684
RvSipReferredByHeaderConstructInMsg() 683
RvSipReferredByHeaderCopy() 685
RvSipReferredByHeaderEncode() 686
RvSipReferredByHeaderFix() 689
RvSipReferredByHeaderGetAddrSpec() 694
RvSipReferredByHeaderGetCidParam() 696
RvSipReferredByHeaderGetCompactForm() 705
RvSipReferredByHeaderGetDispName() 692
RvSipReferredByHeaderGetOtherParams() 698
RvSipReferredByHeaderGetStrBadSyntax() 701
RvSipReferredByHeaderGetStringLength() 691
RvSipReferredByHeaderHandle 936
RvSipReferredByHeaderParse() 687
RvSipReferredByHeaderParseValue() 688
RvSipReferredByHeaderSetAddrSpec() 695
RvSipReferredByHeaderSetCidParam() 697
RvSipReferredByHeaderSetCompactForm() 704
RvSipReferredByHeaderSetDispName() 693
RvSipReferredByHeaderSetOtherParams() 700
RvSipReferredByHeaderSetStrBadSyntax() 703
RvSipReferredByHeaderStringName 970
RvSipReferToHeaderConstruct() 660
RvSipReferToHeaderConstructInMsg() 659
RvSipReferToHeaderCopy() 661
RvSipReferToHeaderEncode() 662
RvSipReferToHeaderFix() 665
RvSipReferToHeaderGetAddrSpec() 668
RvSipReferToHeaderGetCompactForm() 680
RvSipReferToHeaderGetDisplayName 670
RvSipReferToHeaderGetDisplayName() 670
RvSipReferToHeaderGetOtherParams() 674
RvSipReferToHeaderGetReplacesHeader() 672
RvSipReferToHeaderGetStrBadSyntax() 676
RvSipReferToHeaderGetStringLength() 667
RvSipReferToHeaderHandle 937
RvSipReferToHeaderParse() 663
RvSipReferToHeaderParseValue() 664
RvSipReferToHeaderSetAddrSpec() 669

RvSipReferToHeaderSetCompactForm() 679
RvSipReferToHeaderSetDisplayName 671
RvSipReferToHeaderSetDisplayName() 671
RvSipReferToHeaderSetOtherParams() 675
RvSipReferToHeaderSetReplacesHeader() 673
RvSipReferToHeaderSetStrBadSyntax() 678
RvSipReferToHeaderStringName 971
RvSipReplacesEarlyFlagType 1005
RvSipReplacesHeaderHandle 938
RvSipReplacesHeaderStringName 972, 979
RvSipReplyToHeaderHandle 939
RvSipRetryAfterHeaderConstruct() 710
RvSipRetryAfterHeaderConstructInMsg() 709
RvSipRetryAfterHeaderCopy() 711
RvSipRetryAfterHeaderEncode() 712
RvSipRetryAfterHeaderFix() 715
RvSipRetryAfterHeaderGetDateHandle() 721
RvSipRetryAfterHeaderGetDeltaSeconds() 719
RvSipRetryAfterHeaderGetDuration() 723
RvSipRetryAfterHeaderGetFormat() 718
RvSipRetryAfterHeaderGetRetryAfterParams( ) 727
RvSipRetryAfterHeaderGetStrBadSyntax() 730
RvSipRetryAfterHeaderGetStrComment() 725
RvSipRetryAfterHeaderGetStringLength() 717
RvSipRetryAfterHeaderHandle 940
RvSipRetryAfterHeaderParse() 713
RvSipRetryAfterHeaderParseValue() 714
RvSipRetryAfterHeaderSetDate() 722
RvSipRetryAfterHeaderSetDeltaSeconds() 720
RvSipRetryAfterHeaderSetDuration() 724
RvSipRetryAfterHeaderSetRetryAfterParams( ) 729
RvSipRetryAfterHeaderSetStrBadSyntax() 732
RvSipRetryAfterHeaderSetStrComment() 726
RvSipRetryAfterHeaderStringName 973
RvSipRouteHopHeaderConstruct() 736
RvSipRouteHopHeaderConstructInMsg() 735
RvSipRouteHopHeaderCopy() 737
RvSipRouteHopHeaderEncode() 738
RvSipRouteHopHeaderFix() 741
RvSipRouteHopHeaderGetAddrSpec() 746
RvSipRouteHopHeaderGetDisplayName() 744
RvSipRouteHopHeaderGetHeaderType() 751
RvSipRouteHopHeaderGetOtherParams() 748
RvSipRouteHopHeaderGetRpoolString() 753
RvSipRouteHopHeaderGetStrBadSyntax() 755
RvSipRouteHopHeaderGetStringLength() 743
RvSipRouteHopHeaderHandle 941
RvSipRouteHopHeaderParse() 739
RvSipRouteHopHeaderParseValue() 740
RvSipRouteHopHeaderSetAddrSpec() 747
RvSipRouteHopHeaderSetDisplayName() 745
RvSipRouteHopHeaderSetHeaderType() 752
RvSipRouteHopHeaderSetOtherParams() 750
RvSipRouteHopHeaderSetRpoolString() 754
RvSipRouteHopHeaderSetStrBadSyntax() 757
RvSipRouteHopHeaderStringName 974
RvSipRouteHopHeaderType 1006
RvSipRSeqHeaderConstruct() 762
RvSipRSeqHeaderConstructInMsg() 761
RvSipRSeqHeaderCopy() 763
RvSipRSeqHeaderEncode() 764
RvSipRSeqHeaderFix() 767
RvSipRSeqHeaderGetResponseNum() 769
RvSipRSeqHeaderGetStrBadSyntax() 772

RvSipRSeqHeaderGetStringLength() 771
RvSipRSeqHeaderHandle 942
RvSipRSeqHeaderIsResponseNumInitialized() 775
RvSipRSeqHeaderParse() 765
RvSipRSeqHeaderParseValue() 766
RvSipRSeqHeaderSetResponseNum() 770
RvSipRSeqHeaderSetStrBadSyntax() 774
RvSipRSeqHeaderStringName 975
RvSipSessionExpiresHeaderConstruct() 780
RvSipSessionExpiresHeaderConstructInMsg() 779
RvSipSessionExpiresHeaderCopy() 781
RvSipSessionExpiresHeaderEncode() 782
RvSipSessionExpiresHeaderFix() 785
RvSipSessionExpiresHeaderGetCompactForm() 798
RvSipSessionExpiresHeaderGetDeltaSeconds() 788
RvSipSessionExpiresHeaderGetOtherParams() 792
RvSipSessionExpiresHeaderGetRefresherType() 790
RvSipSessionExpiresHeaderGetStrBadSyntax() 794
RvSipSessionExpiresHeaderGetStringLength() 787
RvSipSessionExpiresHeaderHandle 943
RvSipSessionExpiresHeaderParse() 783
RvSipSessionExpiresHeaderParseValue() 784
RvSipSessionExpiresHeaderSetCompactForm() 797
RvSipSessionExpiresHeaderSetDeltaSeconds() 789
RvSipSessionExpiresHeaderSetOtherParams() 793
RvSipSessionExpiresHeaderSetRefresherType() 791
RvSipSessionExpiresHeaderSetStrBadSyntax() 796
RvSipSessionExpiresHeaderStringName 976
RvSipSessionExpiresRefresherType 1007
RvSipSubscriptionReason 1008
RvSipSubscriptionStateHeaderConstruct() 802
RvSipSubscriptionStateHeaderConstructInMsg() 801
RvSipSubscriptionStateHeaderCopy() 803
RvSipSubscriptionStateHeaderEncode() 804
RvSipSubscriptionStateHeaderFix() 807
RvSipSubscriptionStateHeaderGetExpiresParam() 819
RvSipSubscriptionStateHeaderGetOtherParams() 816
RvSipSubscriptionStateHeaderGetReason() 813
RvSipSubscriptionStateHeaderGetRetryAfter() 821
RvSipSubscriptionStateHeaderGetStrBadSyntax() 823
RvSipSubscriptionStateHeaderGetStringLength() 809
RvSipSubscriptionStateHeaderGetStrReason() 814
RvSipSubscriptionStateHeaderGetStrSubstate() 811
RvSipSubscriptionStateHeaderGetSubstate() 810
RvSipSubscriptionStateHeaderHandle 944
RvSipSubscriptionStateHeaderParse() 805
RvSipSubscriptionStateHeaderParseValue() 806
RvSipSubscriptionStateHeaderSetExpiresParam() 820
RvSipSubscriptionStateHeaderSetOtherParams() 818
RvSipSubscriptionStateHeaderSetReason() 815
RvSipSubscriptionStateHeaderSetRetryAfter() 822
RvSipSubscriptionStateHeaderSetStrBadSyntax() 825
RvSipSubscriptionStateHeaderSetSubstate() 812
RvSipSubscriptionStateHeaderStringName 977
RvSipSubscriptionSubstate 1009
RvSipTelUriEnumdiType 1010
RvSipTimestampHeaderHandle 945

RvSipToHeaderConstructInMsg() 607
RvSipTransport 1011
RvSipUrlLrParamType 1014
RvSipUrlLrType 1014
RvSipUrlScheme 1013
RvSipUserParam 1012
RvSipViaHeaderConstruct() 854
RvSipViaHeaderConstructInMsg() 853
RvSipViaHeaderCopy() 855
RvSipViaHeaderEncode() 856
RvSipViaHeaderFix() 859
RvSipViaHeaderGetAliasParam() 893
RvSipViaHeaderGetBranchParam() 878
RvSipViaHeaderGetCompactForm() 895
RvSipViaHeaderGetCompParam() 889
RvSipViaHeaderGetHiddenParam() 880
RvSipViaHeaderGetHost() 866
RvSipViaHeaderGetMaddrParam() 872
RvSipViaHeaderGetOtherParams() 882
RvSipViaHeaderGetPortNum() 868
RvSipViaHeaderGetReceivedParam() 874
RvSipViaHeaderGetRpoolString() 884
RvSipViaHeaderGetRportParam() 876
RvSipViaHeaderGetStrBadSyntax() 886
RvSipViaHeaderGetStrCompParam() 890
RvSipViaHeaderGetStringLength() 862
RvSipViaHeaderGetStrTransport() 864
RvSipViaHeaderGetTransport() 863
RvSipViaHeaderGetTtlNum() 870
RvSipViaHeaderHandle 946
RvSipViaHeaderParse() 857
RvSipViaHeaderParseValue() 858
RvSipViaHeaderSetAliasParam() 892
RvSipViaHeaderSetBranchParam() 879
RvSipViaHeaderSetCompactForm() 894
RvSipViaHeaderSetCompParam() 891
RvSipViaHeaderSetHiddenParam() 881
RvSipViaHeaderSetHost() 867
RvSipViaHeaderSetMaddrParam() 873
RvSipViaHeaderSetOtherParams() 883
RvSipViaHeaderSetPortNum() 869
RvSipViaHeaderSetReceivedParam() 875
RvSipViaHeaderSetRpoolString() 885
RvSipViaHeaderSetRportParam() 877
RvSipViaHeaderSetStrBadSyntax() 888
RvSipViaHeaderSetTransport() 865
RvSipViaHeaderSetTtlNum() 871
RvSipViaHeaderStringName 978
RvSipWarningHeaderHandle 947