SDP Stack

Programmer and Reference Guide

RADVISION

Delivering the Visual Experience

NOTICE

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

RADVISION SDP Stack version 3.1 Programmer and Reference Guide, July 2006

Publication 14

http://www.radvision.com

CONTENTS

About This Manual

The RADVISION SDP Stack is used with the RADVISION SIP Toolkit for the purpose of multimedia session announcement, multimedia session invitation, and other forms of multimedia session initiation. The *RADVISION SDP Stack Programmer and Reference Guide* describes how to use the SDP libraries, and lists the Application Programming Interfaces (API) functions of the SDP Stack. Each function has a general description, syntax description and additional information. Also contained are the enumerations and structure types that are defined in the API.

Related Documentation

The following documentation is provided with the RADVISION SDP Stack:

- *RADVISION SDP Stack Programmer Guide and Reference Guide* (PDF format)
- *RADVISION SDP Stack Online Reference* (HTML and CHM formats)

1

USING THE SDP LIBRARY

INTRODUCTION

SDP (Session Description Protocol—RFC 2327) is the protocol used to describe multimedia session announcement, multimedia session invitation, and other forms of multimedia session initiation. A multimedia session is defined, for these purposes, as a set of media streams that exist for a duration of time. You can find all the public API functions defined by SDP library in the *rvsdp.h* file.

SDP PACKETS

SDP packets usually include the following information:

SESSION INFORMATION

- Session name and purpose
- Time(s) the session is active

Since the resources necessary for participating in a session may be limited, it is useful to include the following additional information:

- Information about the bandwidth to be used by the session
- Contact information for the person responsible for the session

MEDIA INFORMATION

- Type of media, such as video and audio
- Transport protocol, such as RTP/UDP/IP and H.320
- Media format, such as H.261 video and MPEG video

- Multicast address and Transport Port for media (IP multicast session)
- Remote address for media and Transport port for contact address (IP unicast session)

RADVISION SDP STACK

The RADVISION SDP Stack is a stand-alone SDP Library which provides SDP message processing functionality. SDP is a tool for specifying media capabilities, not a communication protocol. The SDP Stack performs the following:

- Parsing and encoding of SDP messages or message parts, such as media descriptions and RTP maps
- Message creation, either from new or existing messages or message parts

API NAMING CONVENTIONS

The API functions in the SDP Library have been defined in a way that allows you to understand and associate functions and their data types.

SYNTAX

Example 1

The following are examples of typical SDP API functions:

```
/*=========================================================*/
RvSdpMsg* rvSdpMsgConstruct(RvSdpMsg * sdp);
/*=========================================================*/
void rvSdpMsgDestruct(RvSdpMsg * sdp);
/*=========================================================*/
RvSdpOrigin rvSdpMsgGetOrigin(cons RvSdpMsg * x);
/*=========================================================*/
RvSdpStatus rvSdpMsgSetSessionName(RvSdpMsg* x,
const char * session_name);
/*=========================================================*/
RvSdpMediaDescr* rvSdpMsgAddMediaDescr(
RvSdpMsg * sdp,RvSdpMediaType media_type,
int port,RvSdpProtocol protocol);
/*=========================================================*/
```

The API syntax consists of types, function names and parameters structured as follows:

```
Rv<TypeName>* rv<TypeName><Verb><Noun>(Rv<TypeName>*
parameter1, parameter2,..)
```

where:

- `Rv<TypeName>` consists of:
  - `Rv`—the namespace that uniquely distinguishes RADVISION-specific TypeNames.
  - `TypeName`—the name of the type, in mixed case.
- `rv<TypeName><Verb><Noun>` is the function name, which consists of:
  - `rv`—an indication that this is a RADVISION-specific function.
  - `TypeName`—the name of the type (in mixed case) with which the function is associated.
  - `Verb`—typically, one of the common function verbs listed in the section, Common Function Verbs.
  - `Noun`—the "operated on" entity. (See below)
- `Rv<TypeName>* parameter1` is the first parameter and usually points to an object of the function's `<TypeName>`. This is also referred to as the "operated on" object.
- `parameter1,..parameterN` are parameters that provide further input to, or output from, the function.

Common Function Verbs

Several common function verbs are used throughout the SDP API, as follows:

- **Construct**—instantiates, or initializes, the "operated on" object. All construct functions (except for functions constructing the whole message object of the RvSdpMsg type) are obsolete and should not be used. "Add" or "Set" functions should be used instead. All construct functions always return the object being constructed, or NULL if the error occurs.
- **ConstructCopy**—constructs an SDP object as a copy of another SDP object with the same type. All ConstructCopy functions (except for functions dealing with the RvSdpMsg type) are obsolete and should not be used. These functions are kept only for backward compatibility.
- **Copy**—copies an SDP object of the same type.
- **Destruct** —tears down an object and frees all allocated resources.

- **Get**—gets an object from the owning object. For example, rvSdpMsgGetOrigin() gets a pointer to an *RvSdpOrigin* object owned by the message.
- **Get**—gets an object by index from the owning object. “Get” is valid only for objects allowing multiple appearances within the owning object. For example, rvSdpMsgGetAttribute() gets an *RvSdpAttribute* object with a specific index within an *RvSdpMsg* object.
- **Get**—gets the field value from an object. For example, rvSdpAttributeGetName() gets the name field of an *RvSdpAttribute* object.
- **Set**—sets the field value within an object. For example, rvSdpAttributeSetName() sets the name field of an *RvSdpAttribute* object.
- **Set**—constructs and sets an owned object within the owning object. For example, rvSdpMsgSetKey() constructs an *RvSdpKey* object within an *RvSdpMsg* object.
- **Add**—constructs and appends an owned object to a list of owned objects. For example, rvSdpMsgAddAttr() constructs a new *RvSdpAttribute* object and will add it to the list of attributes of an *RvSdpMsg* object. These functions are defined only for objects with multiple appearances within the owning object.
- **GetNumOf**—gets the number of owned objects of a specific type within the owning object. For example, rvSdpMediaDescrGetNumOfConnections() gets the number of *RvSdpConnection* objects owned by a specific *RvSdpMediaDescr* object.
- **GetFirst, GetNext**—allows iterating on an object list within an owning object. These functions require an *RvSdpListIter* object.
- **Remove**—removes and destroys an object by index from a list of owning object. For example rvSdpMsgRemoveConnection() removes and destroys an *RvSdpConnection* object that has a specific index within the connections list of an *RvSdpMediaDescr* object.
- **RemoveCurrent**—removes and destroys an object by a specified by *RvSdpListIter* object from the list of owning objects. For example rvSdpMsgRemoveCurrentConnection() will remove and destroy an *RvSdpConnection* object specified by an iterator within the connections list of a *RvSdpMsg* object.

SDP Message Objects

The RADVISION SDP Library provides the objects listed in Table 1-1 for working with SDP messages.

Table 1-1 *Objects for SDP*

Message Objects	Media Descriptors	Time Description	General-purpose Message Parts
RvSdpMsg	*RvSdpMediaDescr*	*RvSdpSessionTime* *RvSdpTimeZoneAdjust* *RvSdpRepeatInterval*	*RvSdpOrigin* *RvSdpEmail* *RvSdpPhone* *RvSdpConnection* *RvSdpBandwidth* *RvSdpKey* *RvSdpAttribute* *RvSdpRtpMap* *RvSdpOther* *RvSdpBadSyntax* *RvSdpKeyMgmtAttr* *RvSdpCryptoAttr*

Figure 1-1 below illustrates the containment relationship between SDP message objects and SDP message parts.

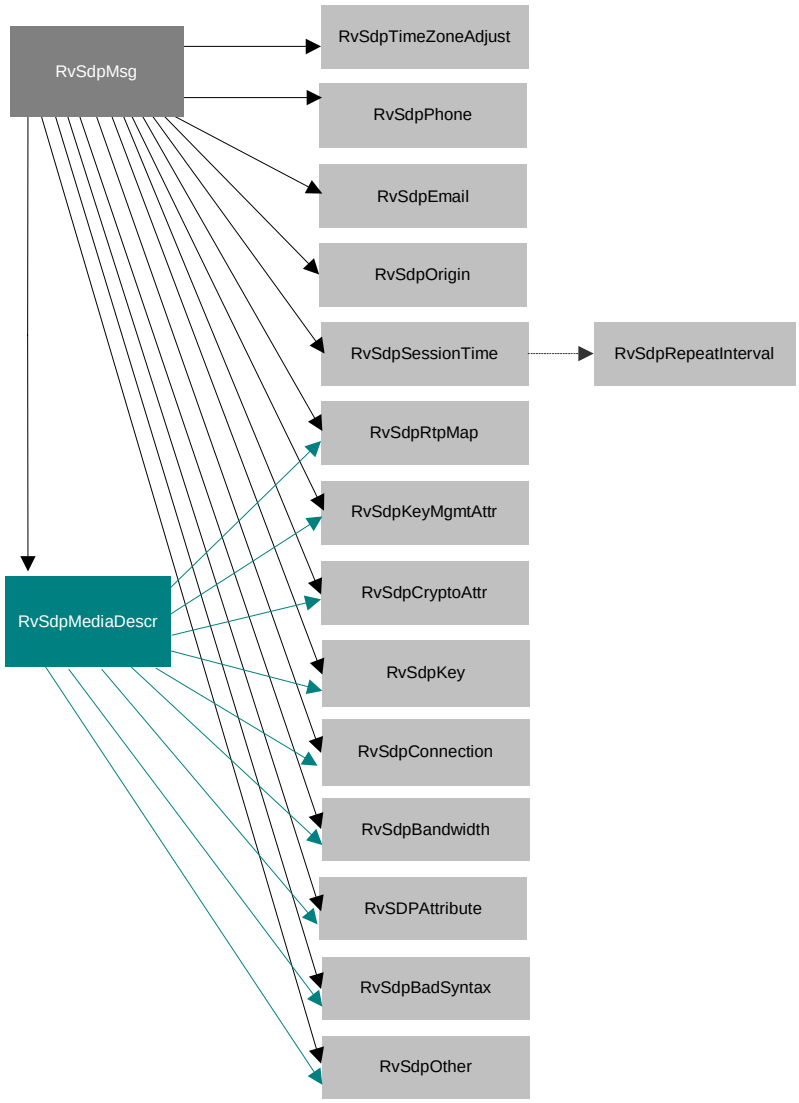

Figure 1-1 *SDP Message Objects*

Reading and Modifying an SDP Message

The *RvSdpMsg* object provides a range of functions for reading and modifying various SDP message parts.

You can fetch message parts by:

- Type
- Name
- Position—you can fetch the following message parts, which can have multiple values, by position:
  - *RvSdpTimeZoneAdjust*
  - *RvSdpMediaDescr*
  - *RvSdpPhone*
  - *RvSdpEmail*
  - *RvSdpSessionTime*
  - *RvSdpRtpMap*
  - *RvSdpKeyMgmtAttr*
  - *RvSdpCryptoAttr*
  - *RvSdpKey*
  - *RvSdpConnection*
  - *RvSdpAttribute*
  - *RvSdpOther*
  - *RvSdpBadSyntax*

Manipulating Messages

The sample code below demonstrates ways of manipulating an SDP message:

Sample Code 1

```
/*=========================================================*/
void ProcessMessage(RvSdpMsg *msg)
{
    /* Checks if this is a non NULL message */

    if (msg!=NULL)
   {
 /* This is a non NULL object - gets the origin field */
 unsigned index;
 RvSdpMediaDescr *media;
 unsigned mediaSize;
```

```
RvSdpOrigin *origin;
char *name="orient";
char *value="portrait";

origin = rvSdpMsgGetOrigin(msg);
/* Do something with origin */

/* Gets all the media descriptor fields in order of
    appearance */
mediaSize = rvSdpMsgGetNumOfMediaDescr(msg);
for (index=0;index< mediaSize; ++index)
{

  media=rvSdpMsgGetMediaDescr(msg, index);
  // Do something with media
}

/* Adds an attribute field at the end of the list */
      rvSdpMsgAddAttr (msg, name, value);
  }
}
/*=========================================================*/
```

CREATING AN SDP MESSAGE

The three ways of creating an SDP message are as follows:

- Construct new
- Construct from an existing SDP message object
- Construct from an encoded message buffer

To create a new SDP message (method 1)

1. Define an *RvSdpMsg* object.
2. Create the message object using rvSdpMsgConstruct(). (This function uses the default allocator.)
3. Set the message fields. See RFC 2327 for a list of mandatory and optional fields.

To create a new SDP message (method 2)

1. Define an *RvAlloc* object.
2. Create the allocator.
3. Define an *RvSdpMsg* object.
4. Create the defined message object using rvSdpMsgConstructA().
5. Set the message fields. See RFC 2327 for a list of mandatory and optional fields.

Sample Code 2

The sample code below demonstrates how to create a a new SDP message:

```
/*=========================================================*/
RvSdpMsg * AppCreateSdpMsgFromNew (void)
{
   /* Creates a SDP message object */
    const char *session_id;
    RvSdpMediaDescr *media;
    int bufLen=200;
    RvSdpStatus eStatus = RV_SDPSTATUS_OK;


    RvSdpMsg *pMsg  = (RvSdpMsg*)malloc(sizeof(RvSdpMsg));
    RvAlloc *pSdpAlloc = (RvAlloc *)malloc(sizeof(RvAlloc));

    rvSdpMsgConstruct(pMsg);

    /* Sets session information */
    eStatus = rvSdpMsgSetSessionInformation(pMsg,"message
    from scratch");
    if (eStatus!=RV_SDPSTATUS_OK)
           return null;

    /* Constructs 'o=' header value (origin)*/
    eStatus = rvSdpMsgSetOrigin(pMsg,"mhandly","12367","0",
    RV_SDPNETTYPE_IN,
```

```
            RV_SDPADDRTYPE_IP4,"126.16.64.4");
            if (eStatus!=RV_SDPSTATUS_OK)
               return null;



/*Gets the origin session ID */
    session_id = rvSdpOriginGetSessionId(pMsg);

    /* Sets 's=' header value */
    eStatus = rvSdpMsgSetSessionName(pMsg,"SDP Seminar");
              if (eStatus!=RV_SDPSTATUS_OK)
                      return null;


    /* Adds other session level headers as needed */

    /* Adds 'm=' header value */
    rvSdpMsgAddMediaDescr(pMsg,RV_SDPMEDIATYPE_AUDIO, 49170,
            RV_SDPPROTOCOL_RTP);

    /* Gets the media descriptor from the SDP message*/
    media = rvSdpMsgGetMediaDescr(pMsg,0);
    rvSdpMediaDescrAddFormat(media,"96");

    /* Adds 'a=' rtpmap value in media description level */
    rvSdpMediaDescrAddRtpMap(media,96, "L8",8000);

    / *Adds 'b=' bandwidth value in media description level
   */
    eStatus = rvSdpMediaDescrSetBandwidth(media,"AS",0);
              if (eStatus!=RV_SDPSTATUS_OK)
                     return null;
```

```
    /* Adds other media level headers as needed */
    return pMsg;

}
/*=========================================================*/
```

To create an SDP message from an existing *RvSdpMsg* object (method 1)

1. Call rvSdpMsgConstructCopy(). (This function uses the default allocator.)
2. Modify the new copy.

To create an SDP message from an existing *RvSdpMsg* object (method 2)

1. Define an *RvSdpMsg* object.
2. Define an *RvAlloc* object.
3. Create the *RvAlloc* object.
4. Call rvSdpMsgConstructCopyA() for the existing *RvSdpMsg* object.
5. Modify the new copy.

Sample Code 3

This sample code demonstrates how to create an *RvSdpMsg* from an existing *RvSdpMsg*:

```
/*=========================================================*/
RvSdpMsg * AppCreateSdpMsgFromExistingMsg(
    IN RvSdpMsg  *pExistingMsg)
{
    RvSdpMsg  *newMsg = (RvSdpMsg*)malloc(sizeof(RvSdpMsg));

    rvSdpMsgConstructCopy (newMsg,pExistingMsg);

    /* Modifies the new message as needed*/
    return newMsg;

}
```

```
/*=======================================================*/
```

To create an *RvSdpMsg* from an encoded message buffer (method 1)

1. Define an *RvSdpMsg* object.
2. Create the defined *message object* using rvSdpMsgConstructParse().

To create an *RvSdpMsg* from an encoded message buffer (method 2)

1. Define an *RvAlloc* object.
2. Create the *RvAlloc* using RvSdpAllocConstruct().
3. Define an *RvSdpMsg* object.
4. Create the defined *RvSdpMsg* using rvSdpMsgConstructParseA().

Sample Code 4

The sample code below demonstrates how to create an *RvSdpMsg* from an encoded message buffer:

```
/*=======================================================*/
RvSdpMsg * AppCreateSdpMsgFromEncodedBuffer(
    char *pBuffer,
    int *pBufSize)
{
    /* Creates an SDP message from an encoded buffer*/

    RvSdpParseStatus eStat;
    RvSdpMsg  *pMsg = (RvSdpMsg*)malloc(sizeof(RvSdpMsg));

    /* Passes the encoded buffer */
    rvSdpMsgConstructParse(pMsg,pBuffer, pBufSize,&eStat);
    if (eStat == RV_SDPPARSER_STOP_ERROR)
        return NULL;
    return pMsg;

}
/*=======================================================*/
```

To create an *RvSdpMsg* from an encoded message buffer while collecting parse warnings (method 3)

1. Define an *RvAlloc* object or use the default allocator.
2. Define a parser data object of the RvSdpParserData type.
3. Construct a *RvSdpParserData* object using rvSdpMsgConstructParserData().
4. Create the *RvAlloc* object using RvSdpAllocConstruct() if it needs to be used.
5. Define an *RvSdpMsg* object.
6. Create the defined *RvSdpMsg* object using rvSdpMsgConstructParse2().
7. Iterate on collected parse warnings using rvSdpParserGetFirstWarning() and rvSdpParserGetNextWarning().
8. The *RvSdpParserData* needs to be destructed at the end using rvSdpMsgDestroyParserData().

Sample Code 5

The sample code below demonstrates how to create an *RvSdpMsg* from an encoded message buffer while collecting parse warnings:

```
/*=========================================================*/
RvSdpMsg * AppCreateSdpMsgFromEncodedBuffer2(
char *pBuffer,
int *pBufSize)
{
/* Creates an SDP message from encoded buffer*/
RvSdpParseStatus eStat;
RvSdpParserData prsData;
RvSdpParserWarningData* prsWarn;
RvSdpMsg *pMsg = (RvSdpMsg*)malloc(sizeof(RvSdpMsg));

/* Constructs a parser data object (using default allocator)
*/
if (rvSdpMsgConstructParserData(&prsData,NULL) == NULL)
{
```

```
   /* failed */
   return NULL;
}

/* Passes the encoded buffer (using default allocator) */
rvSdpMsgConstructParse2(&prsData,pMsg,pBuffer,
pBufSize,&eStat, NULL);
if (eStat == RV_SDPPARSER_STOP_ERROR)
     return NULL;

{
  char txt[128];
  RvSdpListIter iter;
  /* Iterates and prints the parse warnings */
  for (prsWarn = rvSdpParserGetFirstWarning(&prsData,&iter);
prsWarn; prsWarn = rvSdpParserGetNextWarning(&iter))
  {
    /* Fetches the text from the parse warning */
    rvSdpGetWarningText(prsWarn,txt,sizeof(txt)-1);
    printf("The warning: %s\n",txt);
  }
}
/* Destroys parser data when collected parse warnings are no
longer needed */
rvSdpMsgDestroyParserData(&prsData);

return pMsg;
}
/*=========================================================*/
```

Encoding an SDP Message

You get the encoded format of an SDP message object according to the following steps.

To get the encoded format of an *RvSdpMsg* object

1. Allocate a buffer.
2. Call rvSdpMsgEncodeToBuf() on the message you want to encode, passing the buffer as parameter.
3. Check the return code. If the function failed because of a lack of buffer space, try again with a larger buffer.

Sample Code 6

This sample code demonstrates how to encode an *RvSdpMsg* object:

```
RvStatus AppEncodeMsgToBuffer(RvSdpMsg *pMsg,
                              int   bufLen,
                              char  *encodedBuf)
{
    RvSdpStatus eStatus;
    rvSdpMsgEncodeToBuf(pMsg,encodedBuf,bufLen,&eStatus);
    if (eStatus != RV_SDPSTATUS_OK)
    {
        /* Error in rvSdpMsgEncodeToBuf function */
    }
    return eStatus;
}
```

Generic and Special Attribute Treatment

Generic and special attributes are treated differently in the SDP library. The special attributes have dedicated API functions that are defined in the library. Currently, the special attributes are:

- Connection mode (a = sendonly/recvonly/sendrecv/inactive)
- RTP map (a=rtpmap: ...)
- Key management (a=key-mgmt: ...)
- Crypto (a=crypto: ...)
- Framerate (a=framerate: ...)
- Codec format parameters (a=fmtp: ...)

Support of some of the special attributes is enabled only if the correspondent compilation switch defined in the *rvsdpconfig.h* is set. The compilation switches are:

- RV_SDP_KEY_MGMT_ATTR—for key management attributes (*RvSdpKeyMgmtAttr*)
- RV_SDP_CRYPTO_ATTR for crypto attributes (*RvSdpCryptoAttr*)
- RV_SDP_FRAMERATE_ATTR for framerate attributes
- RV_SDP_FMTP_ATTR for codec format parameters (*fmtp*) attributes

For example, if the RV_SDP_KEY_MGMT_ATTR compilation switch is not set (the line, #define RV_SDP_KEY_MGMT_ATTR, is commented in the *rvsdpconfig.h* file) the *RvSdpKeyMgmtAttr* is treated as a generic attribute. In addition, all rvSdpKeyMgmtXXXX(), rvSdpMsgXXXKeyMgmt() and rvSdpMediaDescrXXXKeyMgmt() API functions, and the RvSdpKeyMgmtAttr data type are not defined.

There are two sets of attribute-handling functions. The first set handles only generic attributes, while the second handles both generic and special attributes. For example, the rvSdpMsgGetNumOfAttr() function will get only the number of generic attributes defined in the context of an *RvSdpMsg*. All special attributes defined above will not be counted by this function. The rvSdpMsgGetNumOfAttr2() function will get the number of all attributes (generic and special) defined in the context of an *RvSdpMsg*. Both functions will not count the attributes defined in the context of one of the media descriptors defined in the message. Special and generic attributes functions add the number “2” to the name of generic-only attribute functions.

Errors and Proprietary Information

This section explains how the SDP Parser supplies the application with information about non-standard SDP lines and how the user can correct and re parse these lines. It also describes how the user can use the SDP package to send proprietary information in an SDP message.

SDP Parser Error Handling

The SDP parser ignores white spaces between tokens. It also handles excessive blank lines before an SDP message but does not handle excessive blank lines in the middle of the message. The SDP parser also does not handle excessive blank lines at the end of the message

When the SDP parser receives an SDP message, it handles three kinds of syntax errors, as follows:

- Unknown tag, meaning an SDP line with the format, tag = value, where *tag* is any character not used by the SDP standard, and *value* is a free text.
- Line with unknown or proprietary format (free text line)
- Standard SDP line with syntax errors or proprietary information.

The handling of syntax errors and proprietary formatted lines in SDP depends on the RV_SDP_CHECK_BAD_SYNTAX compilation switch defined in the *rvsdpconfig.h* file. If this switch is disabled (the correspondent line in *rvsdpconfig.h* is commented) the parsing will fail for non-standard SDP input. No "Other" or "Bad Syntax" objects are defined in the message if RV_SDP_CHECK_BAD_SYNTAX is disabled.

Handling of Unknown Tag Lines

Each *RvSdpMsg* and *RvSdpMediaDescr* object contains a list of *RvSdpOther* objects. Each *RvSdpOther* object represents a line of unknown tag. When the parser recognizes such a line, it constructs an object of the RvSdpOther type, where the tag is the unknown tag, and the value is the text after the equal (=) sign. The parser adds the new object to the list of *RvSdpOther* objects of the message or media descriptor according to the order of the lines.

Handling of Unknown Format Lines

Handling unknown format lines is done in the same manner as handling unknown tag lines. The difference is that when the line is not of the "x=text" format, the value of the tag in the RvSdpOther structure is zero (0).

Handling a Regular SDP Line with Syntax Errors

Every message part that has a constrained syntax, and may have syntax errors, contains a Bad Syntax field.

Upon encountering a standard SDP line with a syntax error or proprietary information, the parser constructs the appropriate object as a Bad Syntax SDP object (*RvSdpEmail*, *RvSdpPhone*, *RvSdpBandwidth*, and so on). The parser places the SDP line contents into the Bad Syntax field of the SDP object. This is the only field that will be set. The Bad Syntax field will contain the raw value of the line—this line will not be parsed.

Every SDP object can be checked if it is a BadSyntax SDP object (if its Bad Syntax field is set) using the rvSdpXXXIsBadSynatx() function (where *XXX* is the SDP object name, such as MediaDescr). The Bad Syntax line can be retrieved by the rvSdpXXXGetBadSyntax() function.

Example

If the SDP message contains the line:

```
c=IN IP4 127.0.0.1 connection 7
```

The rvSdpConnectionIsBadSyntax() function will return TRUE, and the rvSdpConnectionGetBadSyntax() function will return
`IN IP4 127.0.0.1 connection 7`

RvSdpBadSyntax List of Objects

Each media descriptor or message objects contains a list of *rvSdpBadSyntax* objects. These objects wrap all the SDP objects that have syntax errors or contain proprietary information. Each *rvSdpBadSyntax* object consists of a type and a field that is a pointer to the SDP object that it wraps.

For each *rvSdpBadSyntax* object, the type of the object it wraps can be retrieved using rvSdpBadSyntaxGetFieldType(), and the wrapped object can be retrieved using rvSdpBadSyntaxGetField().

The field types are defined with the RvSdpFieldTypes enumeration. The rvSdpBadSyntaxGetField() function returns a pointer of the RvSdpGenericFieldPtr type that can be cast to each of the line objects.

When the parser handles a standard SDP line with a syntax error, it will construct a new *rvSdpBadSyntax* object and will use it to wrap the SDP object. The *rvSdpBadSyntax* object will contain the SDP object type and a pointer to the SDP object. The parser will add the *rvSdpBadSyntax* object to the media descriptor or message Bad Syntax list of the message. The object will appear in the list according to the order in the SDP message.

The list is manipulated as every other SDP list object using the following functions:

- rvSdpMsgGetNumOfBadSyntax()
- rvSdpMsgGetNumOfBadSyntax2()
- rvSdpMsgGetFirstBadSyntax()
- rvSdpMsgGetNextBadSyntax()
- rvSdpMsgGetBadSyntax()
- rvSdpMediaDescrGetNumOfBadSyntax()
- rvSdpMediaDescrGetFirstBadSyntax()
- rvSdpMediaDescrGetNextBadSyntax()
- rvSdpMediaDescrGetBadSyntax()

The rvSdpMsgGetNumOfBadSyntax2() function gets the number of *rvSdpBadSyntax* objects in the *RvSdpMsg* including the *rvSdpBadSyntax* objects appearing in one of the *RvSdpMediaDescr* objects contained in the *RvSdpMsg*. The rvSdpMsgGetNumOfBadSyntax() function gets the number of *rvSdpBadSyntax* objects in the message only.

The user of the SDP Stack can approach a *rvSdpBadSyntax* object in two ways.

- Get the line object, and check if the BadSyntaxField is set.
- Go through the message or media descriptor and check each member of its *rvSdpBadSyntax* object list.

Correcting an SDP object with the BadSyntax field Set

The first step for correcting an SDP object with a set Bad Syntax field is changing the Bad Syntax. This should be done using the rvSdpConnectionSetBadSyntax() API function.

The next step is to use the rvSdpXXXReparse() API function. This function parses the Bad Syntax field string into the SDP object. The function also removes the SDP object from the Bad Syntax object list of the message or of the *RvSdpMediaDescr.*

If the reparse succeeds, the Bad Syntax field of the object will not be set and the object will not appear in the Bad Syntax list of the message or of the *RvSdpMediaDescr.* If the reparse fails, the Bad Syntax field will be set and the object will appear in the Bad Syntax object list.

Example

```
if (rvSdpConnectionReparse(msg, conn,&tempLen,&stat,msg-
>alloc) == NULL)
       printf("reparse failed\n");
```

Sending Proprietary Information in SDP Messages

Proprietary information can be divided into the same categories as the syntax errors:

- Proprietary tag, meaning an SDP line with the format, tag = value, where *tag* is any character not used by the SDP standard, and *value* is a free text.
- Line with proprietary format (free text line)
- Regular line with proprietary information.

Sending a Line with Proprietary Tag

The user should construct an *RvSdpOther* object with the proprietary tag and value and then add it to the message or *RvSdpMediaDescr.* (The *RvSdpOther* objects of the same message or the *RvSdpMediaDescr* will be encoded and sent according to the order they were added to the *RvSdpMsg* or *RvSdpMediaDescr*).

Sending a Line with Proprietary Format (Free Text Line)

The user should construct an *RvSdpOther* object with the zero tag and proprietary value and then add it to the message or *RvSdpMediaDescr*. (The *RvSdpOther* objects of the same message or *RvSdpMediaDescr* will be encoded and sent according to the order they were added to the *RvSdpMsg* or *RvSdpMediaDescr*).

Standard SDP line with Proprietary Information or Format

The application using the SDP Stack should construct an SDP object with its Bad Syntax field set to the standard SDP line with the proprietary information, using the rvSdpMsgSetBadSyntaxXXX(), rvSdpMsgAddBadSyntaxXXX, rvSdpMediaDescrSetBadSyntaxXXX() or rvSdpMediaDescrAddBadSyntaxXXX function, and add it to the *RvSdpMsg* or *RvSdpMediaDescr*.

SDP Compilation Switches

The SDP library uses a number of compilation switches which are listed below. These switches are defined in the *rvsdpconfig.h* file.

- RV_SDP_ADS_IS_USED—can be set if the SDP library is used with the ADS library. Setting this switch enables the allocator construction/destruction functions.
- RV_SDP_KEY_MGMT_ATTR—enables/disables API functions related to the *RvSdpKeyMgmtAttr*.
- RV_SDP_CRYPTO_ATTR—enables/disables API functions related to the *RvSdpCryptoAttr*.
- RV_SDP_FRAMERATE_ATTR—enables/disables API functions related to the framerate special attribute.
- RV_SDP_FMTP_ATTR—enables/disables API functions related to the codec format parameters (*fmtp*) special attribute.
- RV_SDP_CHECK_BAD_SYNTAX—enables/disables the handling of non-standard formats in SDP messages.
- RV_SDP_ENABLE_REPARSE—enables/disables SDP object reparse functions.
- RV_SDP_USE_MACROS—if set, many of the SDP functions become macros. Setting this switch reduces the library footprint, but makes the debug process more complicated. It is recommended to disable this switch when in the debug stage and to enable it for the release version.
- RV_SDP_LOG_FILE_NAME—defines the name of log file created by the library. See RvSdpMgrConstructWithConfig() API function for more information on the logging mechanism of the SDP library.

2

SDP Library Construction Functions

What's in this Chapter

This section contains the functions used for the construction and destruction of the SDP library. It also contains functions for constructing and destructing an SDP allocator (*RvAlloc*). However, these functions can be used only if the SDP library is used with the ADS module. Some of the functions defined in this section are present only if the RV_SDP_ADS_IS_USED compilation switch defined in the *rvsdpconfig.h* file is set.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- RvSdpAllocConstruct()
- RvSdpAllocDestruct()
- RvSdpMgrConstructWithConfig()
- RvSdpMgrConstruct()
- RvSdpMgrDestruct()

RvSdpAllocConstruct()

DESCRIPTION

Constructs an *RvAlloc*. An *RvAlloc* is used whenever you need to allocate space from RPOOL. This function is defined only if the RV_SDP_ADS_IS_USED compilation switch is enabled.

SYNTAX

```
RvStatus RvSdpAllocConstruct(
    HRPOOL      hPool,
    RvAlloc*    sdpAlloc);
```

PARAMETERS

hPool

The pool that the SDP library will use.

sdpAlloc

A pointer to the initialized *RvAlloc*. This parameter must point to valid memory.

RETURN VALUES

Returns RV_OK if the function succeeds or an error code if the function fails.

RvSdpAllocDestruct()

DESCRIPTION

Destroys an *RvAlloc*. This function is called after an *RvSdpMsg* is destroyed. This function is defined only if the RV_SDP_ADS_IS_USED compilation switch is enabled.

SYNTAX

```
void RvSdpAllocDestruct(
    RvAlloc*    sdpAlloc);
```

PARAMETERS

sdpAlloc

An *RvAlloc* of the message.

RETURN VALUES

None.

RvSdpMgrConstructWithConfig()

DESCRIPTION

Constructs an SDP library and defines its logging behavior. This function has to be called prior to any other SDP API call. There are four ways to define the logging behavior of the SDP library, as follows:

- When *pStackConfig* is NULL. This is the default logging behavior. The SDP will create a log file named as defined by RV_SDP_LOG_FILE_NAME, which appears in the *rvsdpconfig.h* file. This file will be used for outputting parsing error messages.
- When *pStackConfig* is not NULL and *pStackConfig->disableSdpLogs* is set to RV_TRUE. The SDP module will not produce log messages.
- When *pStackConfig* is not NULL and *pStackConfig->logManagerPtr* is set to the log handle of another RADVISION module. In this case, the log messages produced by the SDP module will be printed by the other RADVISION module. When SDP is used with the SIP Stack Toolkit, use the RvSipStackGetLogHandle() SIP API function to get the log handle of the SIP module.
- When *pStackConfig* is not NULL and *pStackConfig->pfnPrintLogEntryEvHandler* is set. In this case, the application-supplied callback (*pfnPrintLogEntryEvHandler*) will be called each time the log message has to be printed. The pfnPrintLogEntryEvHandler callback will be called with *logContext* as a function argument.

SYNTAX

```
RvStatus RvSdpMgrConstructWithConfig(
    RvSdpStackCfg    *pStackConfig,
    RvUint32         sizeOfCfg);
```

PARAMETERS

pStackCfg

The structure containing the SDP Stack configuration parameters.

sizeOfCfg

The size of the configuration structure.

RETURN VALUES

Returns RV_OK if the function succeeds, or an error code if the function fails.

RvSdpMgrConstruct()

DESCRIPTION

Constructs an SDP library with default logging behavior. This function has to be called prior to any other SDP API call. See RvSdpMgrConstructWithConfig() for a description of the default logging behavior.

SYNTAX

```
RvStatus RvSdpMgrConstruct(void);
```

PARAMETERS

None.

RETURN VALUES

Returns RV_OK if the function succeeds, or an error code if the function fails.

RvSdpMgrDestruct()

DESCRIPTION

Destructs an SDP library. SDP API functions cannot be called after the library is destructed.

SYNTAX

```
void RvSdpMgrDestruct(void);
```

PARAMETERS

None.

RETURN VALUES

None.

3

MESSAGE PARSE AND ENCODE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used for message parsing and encoding. Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMsgConstructParse()
- rvSdpMsgConstructParseA()
- rvSdpMsgConstructParserData()
- rvSdpMsgDestroyParserData()
- rvSdpMsgConstructParse2()
- rvSdpMsgEncodeToBuf()

rvSdpMsgConstructParse()

DESCRIPTION

Parses an SDP text message and constructs an *RvSdpMsg* from the SDP text message.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructParse(
    RvSdpMsg*            msg,
    char*                txt,
    int*                 len,
    RvSdpParseStatus*    stat);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* to be constructed. This parameter must point to valid memory.

txt

Contains a pointer to the beginning of the SDP text message.

len

The length of the parsed *RvSdpMsg*.

stat

The status of the parsing termination. This parameter will be RV_SDPPARSER_STOP_ERROR if there was a parsing error, otherwise it will have another status specifying a legal character that caused the parser to stop. For more information on RvSdpParseStatus, see the Enumerations chapter.

RETURN VALUES

Returns a pointer to the new *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgConstructParseA()

DESCRIPTION

Parses an SDP text message and constructs an *RvSdpMsg* from the SDP text message using the provided *RvAlloc*.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructParseA(
    RvSdpMsg*            msg,
    char*                txt,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* to be constructed. This parameter must point to valid memory.

txt

Contains a pointer to the beginning of the SDP text message.

len

The length of the parsed *RvSdpMsg*.

stat

The status of the parsing termination. This parameter will be RV_SDPPARSER_STOP_ERROR if there was a parsing error, otherwise it will have another status specifying a legal character that caused the parser to stop. For more information on RvSdpParseStatus, see the Enumerations chapter.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the new *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgConstructParserData()

DESCRIPTION

Constructs an *RvSdpParserData*. This parser data is used in further calls to rvSdpMsgConstructParse2(). The *RvSdpParserData* has to be destroyed with rvSdpMsgDestroyParserData() API function.

SYNTAX

```
RvSdpParserData* rvSdpMsgConstructParserData(
    RvSdpParserData    *pD,
    RvAlloc            *a);
```

PARAMETERS

pD

A pointer to the *RvSdpParserData*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpParserData*, or NULL if the function fails.

rvSdpMsgDestroyParserData()

DESCRIPTION

Destroys the *RvSdpParserData* and frees all internal memory.

SYNTAX

```
void rvSdpMsgDestroyParserData(
    RvSdpParserData    *pD);
```

PARAMETERS

pD

A pointer to the *RvSdpParserData* to be destructed.

RETURN VALUES

None.

rvSdpMsgConstructParse2()

DESCRIPTION

Parses an SDP text message and constructs an *RvSdpMsg* from the SDP text message. This function also collects parse warnings during the parse process.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructParse2(
    RvSdpParserData*     pD,
    RvSdpMsg*            msg,
    char*                txt,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

pD

A pointer to the constructed *RvSdpParserData* to be used during the parsing.

msg

A pointer to the *RvSdpMsg* to be constructed. This parameter must point to valid memory.

txt

Contains a pointer to the beginning of the SDP text message.

len

The length of the parsed *RvSdpMsg*.

stat

The status of parsing termination. This parameter will be RV_SDPPARSER_STOP_ERROR if there was a parsing error, otherwise it will be another status specifying a legal character that caused the parser to stop. For more information on RvSdpParseStatus, see the Enumerations chapter.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the new *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgEncodeToBuf()

DESCRIPTION

Takes an *RvSdpMsg* as input and encodes it as text into a buffer (according to the SDP syntax).

SYNTAX

```
char* rvSdpMsgEncodeToBuf(
    RvSdpMsg*       msg,
    char*           buf,
    int             len,
    RvSdpStatus*    stat);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* to be encoded.

buf

A pointer to a buffer where the message will be encoded.

len

The length of the buffer.

stat

A pointer to a variable where the status of the encoding will be set. If encoding was successful, this parameter will be set to an RV_OK value. Otherwise, one of the error values will be used.

RETURN VALUES

Returns a pointer past the end of the encoding (buf + number of encoded bytes), or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpParserGetFirstWarning()
- rvSdpParserGetNextWarning()
- rvSdpGetWarningText()

rvSdpParserGetFirstWarning()

DESCRIPTION

Returns the first warning object in the warnings list of parser data. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpParserWarningData* rvSdpParserGetFirstWarning(
    RvSdpParserData*    pD,
    RvSdpListIter*      iter);
```

PARAMETERS

pD

A pointer to the *RvSdpParserData*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpParserGetNextWarning() calls.

RETURN VALUES

Returns a pointer to the *RvSdpParserWarningData*, or to a NULL pointer if there are no warnings in the list.

rvSdpParserGetNextWarning()

DESCRIPTION

Returns the next *RvSdpParserWarningData* in the warnings list of parser data.

SYNTAX

```
RvSdpParserWarningData* rvSdpParserGetNextWarning(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rvSdpParserGetFirstWarning() or rvSdpParserGetNextWarning().

RETURN VALUES

Returns a pointer to the *RvSdpParserWarningData*, or to a NULL pointer if there are no more warnings in the list.

rvSdpGetWarningText()

DESCRIPTION

Gets the text from the parse warning.

SYNTAX

```
void rvSdpGetWarningText(
    RvSdpParserWarningData*    wd,
    char                       *txt,
    int                        len);
```

PARAMETERS

wd

A pointer to a valid *RvSdpParserWarningData*.

txt

The buffer to be used for the parse warning text.

len

The length of the *txt* buffer.

RETURN VALUES

None.

4

MESSAGE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions for operating on *RvSdpMsg* objects. Please note that no matter how an *RvSdpMsg* was constructed, it has to be destructed with RvSdpAllocDestruct().

Included in this section are:

- Control Functions
- Get/Set Functions

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMsgConstruct()
- rvSdpMsgConstructA()
- rvSdpMsgConstructCopy()
- rvSdpMsgConstructCopyA()
- rvSdpMsgDestruct()
- rvSdpMsgCopy()
- rvSdpMsgDestroyVersion()
- rvSdpMsgCopySdpVersion()
- rvSdpMsgDestroySessionName()
- rvSdpMsgCopySdpSessionId()
- rvSdpMsgDestroyInformation()
- rvSdpMsgDestroyUri()
- rvSdpMsgCopyURI()
- rvSdpMsgUriIsBadSyntax()
- rvSdpMsgAddEmail()
- rvSdpMsgAddBadSyntaxEmail()
- rvSdpMsgRemoveCurrentEmail()
- rvSdpMsgRemoveEmail()
- rvSdpMsgClearEmail()
- rvSdpMsgAddPhone()
- rvSdpMsgAddBadSyntaxPhone()
- rvSdpMsgRemoveCurrentPhone()
- rvSdpMsgRemovePhone()
- rvSdpMsgClearPhones()
- rvSdpMsgAddConnection()
- rvSdpMsgRemoveCurrentConnection()
- rvSdpMsgRemoveConnection()
- rvSdpMsgClearConnection()
- rvSdpMsgAddBandwidth()
- rvSdpMsgRemoveCurrentBandwidth()
- rvSdpMsgRemoveBandwidth()
- rvSdpMsgClearBandwidth()

- rvSdpMsgAddSessionTime()
- rvSdpMsgAddBadSyntaxSessionTime()
- rvSdpMsgRemoveCurrentSessionTime()
- rvSdpMsgRemoveSessionTime()
- rvSdpMsgClearSessionTime()
- rvSdpMsgTZACopy()
- rvSdpMsgTZADestroy()
- rvSdpMsgTZAIsUsed()
- rvSdpMsgIsBadSyntaxZoneAdjustment()
- rvSdpMsgTimeAddZoneAdjustment()
- rvSdpMsgRemoveCurrentZoneAdjustment()
- rvSdpMsgRemoveTimeZoneAdjust()
- rvSdpMsgClearZoneAdjustment()
- rvSdpMsgAddAttr()
- rvSdpMsgRemoveCurrentAttribute()
- rvSdpMsgRemoveAttribute()
- rvSdpMsgClearAttr()
- rvSdpMsgRemoveCurrentAttribute2()
- rvSdpMsgRemoveAttribute2()
- rvSdpMsgClearAttr2()
- rvSdpMsgAddRtpMap()
- rvSdpMsgAddBadSyntaxRtpMap()
- rvSdpMsgRemoveCurrentRtpMap()
- rvSdpMsgRemoveRtpMap()
- RvSdpMsgClearRtpMap()
- rvSdpMsgAddKeyMgmt()
- rvSdpMsgAddBadSyntaxKeyMgmt()
- rvSdpMsgRemoveCurrentKeyMgmt()
- rvSdpMsgRemoveKeyMgmt()
- rvSdpMsgClearKeyMgmt()
- rvSdpMsgAddCrypto()
- rvSdpMsgAddBadSyntaxCrypto()
- rvSdpMsgRemoveCurrentCrypto()

- rvSdpMsgRemoveCrypto()
- rvSdpMsgClearCrypto()
- rvSdpMsgAddMediaDescr()
- rvSdpMsgInsertMediaDescr()
- rvSdpMsgAddBadSyntaxMediaDescr()
- rvSdpMsgRemoveCurrentMediaDescr()
- rvSdpMsgRemoveMediaDescr()
- rvSdpMsgClearMediaDescr()
- rvSdpMsgAddOther()
- rvSdpMsgRemoveCurrentOther()
- rvSdpMsgRemoveOther()
- rvSdpMsgClearOther()

rvSdpMsgConstruct()

DESCRIPTION

Constructs an instance of an *RvSdpMsg*, initializes all internal fields, and allocates memory for the buffer of the string and pools of reusable objects.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstruct(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* instance to be constructed. If the value is NULL, the instance will be allocated within the function.

RETURN VALUES

Returns a valid *RvSdpMsg* pointer on success, or NULL on failure.

rvSdpMsgConstructA()

DESCRIPTION

Constructs an instance of an *RvSdpMsg*, initializes all internal fields, and allocates memory for the buffer of the string and pools of reusable objects.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructA(
    RvSdpMsg*    msg,
    RvAlloc*     a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* instance to be constructed. If the value is NULL, the instance will be allocated within the function.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a valid *RvSdpMsg* pointer on success, or NULL on failure.

rvSdpMsgConstructCopy()

DESCRIPTION

Copies the instance of an *RvSdpMsg* from source to destination. The destination *RvSdpMsg* will be constructed. If the destination *RvSdpMsg* is NULL, a pointer to the destination *RvSdpMsg* will be allocated within the function.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructCopy(
    RvSdpMsg*          dest,
    const RvSdpMsg*    src);
```

PARAMETERS

dest

A pointer to a valid *RvSdpMsg*, or NULL.

src

A pointer to the source *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the input *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgConstructCopyA()

DESCRIPTION

Copies the instance of an *RvSdpMsg* from source to destination. The destination *RvSdpMsg* will be constructed. If the destination *RvSdpMsg* is NULL, a pointer the destination *RvSdpMsg* will be allocated within the function.

SYNTAX

```
RvSdpMsg* rvSdpMsgConstructCopyA(
    RvSdpMsg*          dest,
    const RvSdpMsg*    src,
    RvAlloc*           a);
```

PARAMETERS

dest

A pointer to a valid *RvSdpMsg*, or NULL.

src

A pointer to the source *RvSdpMsg*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the input *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgDestruct()

DESCRIPTION

Destroys an *RvSdpMsg* and frees all internal memory.

SYNTAX

```
void rvSdpMsgDestruct(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*t to be destructed.

RETURN VALUES

None.

rvSdpMsgCopy()

DESCRIPTION

Copies an instance of an *RvSdpMsg* from source to destination. The destination *RvSdpMsg* must be constructed prior to calling this function.

SYNTAX

```
RvSdpMsg* rvSdpMsgCopy(
    RvSdpMsg*          dest,
    const RvSdpMsg*    src);
```

PARAMETERS

dest

A pointer to the constructed *RvSdpMsg*.

src

A pointer to the source *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the input *RvSdpMsg*, or NULL if the function fails.

rvSdpMsgDestroyVersion()

DESCRIPTION

Destroys the version field of the *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgDestroyVersion(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgCopySdpVersion()

DESCRIPTION

Sets the version field of *dstMsg* in an *RvSdpMsg* to that of *src*.

SYNTAX

```
RvSdpStatus rvSdpMsgCopySdpVersion(
    const RvSdpVersion*    src,
    RvSdpMsg*              dstMsg);
```

PARAMETERS

dstMsg

A pointer to the *RvSdpMsg*.

src

The source SDP version.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgDestroySessionName()

DESCRIPTION

Destroys the session name field of the *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgDestroySessionName(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgCopySdpSessionId()

DESCRIPTION

Sets the session name field of *dstMsg* in an *RvSdpMsg* to that of *src*.

SYNTAX

```
RvSdpStatus rvSdpMsgCopySdpSessionId(
    const RvSdpSessId*    src,
    RvSdpMsg*             dstMsg);
```

PARAMETERS

dstMsg

A pointer to the *RvSdpMsg*.

src

The source of the SDP session name.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgDestroyInformation()

DESCRIPTION

Destroys the information field of the *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgDestroyInformation(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgDestroyUri()

DESCRIPTION

Destroys the URI field of the *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgDestroyUri(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgCopyURI()

DESCRIPTION

Sets the URI field of *dstMsg* in an *RvSdpMsg* to that of *src*.

SYNTAX

```
RvSdpStatus rvSdpMsgCopyURI(
    const RvSdpUri*    src,
    RvSdpMsg*          dstMsg);
```

PARAMETERS

dstMsg

A pointer to the *RvSdpMsg*.

src

The source SDP URI.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgUriIsBadSyntax()

DESCRIPTION

Indicates whether or not the URI field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpMsgUriIsBadSyntax(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpMsgAddEmail()

DESCRIPTION

Adds a new *RvSdpEmail* at the session level.

SYNTAX

```
RvSdpEmail* rvSdpMsgAddEmail(
    RvSdpMsg*       msg,
    const char*     email_addr,
    const char*     string);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

email_addr

The new email address.

string

Optional free text. This parameter is set to a string of zero length ("") if not required.

RETURN VALUES

Returns a pointer to the newly created *RvSdpEmail* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxEmail()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpEmail* at the session level.

SYNTAX

```
RvSdpEmail* rvSdpMsgAddBadSyntaxEmail(
    RvSdpMsg*       msg,
    const char*     text);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

text

The proprietary value of the *RvSdpEmail* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpEmail* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentEmail()

DESCRIPTION

Removes (and destructs) an *RvSdpEmail* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentEmail(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstEmail() or rvSdpMsgGetNextEmail() function.

RETURN VALUES

None.

rvSdpMsgRemoveEmail()

DESCRIPTION

Removes (and destructs) an *RvSdpEmail* by index.

SYNTAX

```
void rvSdpMsgRemoveEmail(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfEmail().

RETURN VALUES

None.

rvSdpMsgClearEmail()

DESCRIPTION

Removes (and destructs) all *RvSdpEmail* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearEmail(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddPhone()

DESCRIPTION

Adds a new *RvSdpPhone* at the session level.

SYNTAX

```
RvSdpPhone* rvSdpMsgAddPhone(
    RvSdpMsg*       msg,
    const char*     phone,
    const char*     string);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

phone

The new phone number.

string

Optional free text. This parameter is set to a string of zero length ("") if not required.

RETURN VALUES

Returns a pointer to the newly created *RvSdpPhone* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxPhone()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpPhone* at the session level.

SYNTAX

```
RvSdpPhone* rvSdpMsgAddBadSyntaxPhone(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of the *RvSdpPhone* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpPhone* if the function succeeds, or NULL pointer if the function fails.

rvSdpMsgRemoveCurrentPhone()

DESCRIPTION

Removes (and destructs) an *RvSdpPhone* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentPhone(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstPhone() or rvSdpMsgGetNextPhone() function.

RETURN VALUES

None.

rvSdpMsgRemovePhone()

DESCRIPTION

Removes (and destructs) an *RvSdpPhone* by index.

SYNTAX

```
void rvSdpMsgRemovePhone(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfPhones().

RETURN VALUES

None.

rvSdpMsgClearPhones()

DESCRIPTION

Removes (and destructs) all *RvSdpPhone* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearPhones(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddConnection()

DESCRIPTION

Adds a new *RvSdpConnection* at the session level.

SYNTAX

```
RvSdpConnection* rvSdpMsgAddConnection(
    RvSdpMsg*        msg,
    RvSdpNetType     net_type,
    RvSdpAddrType    addr_type,
    const char*      addr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

net_type

The network type.

addr_type

The address type.

addr

The address, depending on the network type. For example, an IP address for an IP network, and so on.

RETURN VALUES

Returns a pointer to the newly created *RvSdpConnection* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentConnection()

DESCRIPTION

Removes (and destructs) an *RvSdpConnection* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentConnection(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rvSdpMsgGetFirstConnection() or rvSdpMsgGetNextConnection() function.

RETURN VALUES

None.

rvSdpMsgRemoveConnection()

DESCRIPTION

Removes (and destructs) an *RvSdpConnection* by index.

SYNTAX

```
void rvSdpMsgRemoveConnection(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfConnections().

RETURN VALUES

None.

rvSdpMsgClearConnection()

DESCRIPTION

Removes (and destructs) all *RvSdpConnection* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearConnection(
    RvSdpMsg*    descr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddBandwidth()

DESCRIPTION

Adds a new *RvSdpBandwidth* at the session level.

SYNTAX

```
RvSdpBandwidth* rvSdpMsgAddBandwidth(
    RvSdpMsg*       msg,
    const char*     bwtype,
    int             b);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

bwtype

The *RvSdpBandwidth* type, such as Conference Total (CT) or Application-Specific Maximum (AS).

b

The *RvSdpBandwidth* value in kilobits per second (kbps).

RETURN VALUES

Returns a pointer to the newly created *RvSdpBandwidth* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentBandwidth()

DESCRIPTION

Removes (and destructs) an *RvSdpBandwidth* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentBandwidth(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rvSdpMsgGetFirstBandwidth() or rvSdpMsgGetNextBandwidth().

RETURN VALUES

None.

rvSdpMsgRemoveBandwidth()

DESCRIPTION

Removes (and destructs) an *RvSdpBandwidth* by index.

SYNTAX

```
void rvSdpMsgRemoveBandwidth(
    RvSdpMsg*    descr,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfBandwidth().

RETURN VALUES

None.

rvSdpMsgClearBandwidth()

DESCRIPTION

Removes (and destructs) all *RvSdpBandwidth* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearBandwidth(
    RvSdpMsg*    descr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddSessionTime()

DESCRIPTION

Adds a new *RvSdpSessionTime*.

SYNTAX

```
RvSdpSessionTime* rvSdpMsgAddSessionTime(
    RvSdpMsg*    msg,
    RvUint32     start,
    RvUint32     stop);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

start

The start time of the SDP session.

stop

The end time of the SDP session.

RETURN VALUES

Returns a pointer to the newly created *RvSdpSessionTime* if the function succeeds, or NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxSessionTime()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpSessionTime* at the session level.

SYNTAX

```
RvSdpSessionTime* rvSdpMsgAddBadSyntaxSessionTime(
    RvSdpMsg*     msg,
    const char    *badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of *RvSdpSessionTime* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpSessionTime* if the function succeeds, or NULL pointer if the function fails.

rvSdpMsgRemoveCurrentSessionTime()

DESCRIPTION

Removes (and destructs) an *RvSdpSessionTime* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentSessionTime(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* set or modified by a previous, successful call to the rvSdpMsgGetFirstSessionTime() or rvSdpMsgGetNextSessionTime() function.

RETURN VALUES

None.

rvSdpMsgRemoveSessionTime()

DESCRIPTION

Removes (and destructs) an *RvSdpSessionTime* by index.

SYNTAX

```
void rvSdpMsgRemoveSessionTime(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfSessionTime().

RETURN VALUES

None.

rvSdpMsgClearSessionTime()

DESCRIPTION

Removes (and destructs) all *RvSdpSessionTime* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearSessionTime(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgTZACopy()

DESCRIPTION

Sets an *RvSdpTimeZoneAdjust* field of a destination *RvSdpMsg* as the *RvSdpTimeZoneAdjust* field of a source *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgTZACopy(
    RvSdpMsg*          dest,
    const RvSdpMsg*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpMsg*.

src

A pointer to the source *RvSdpMsg*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgTZADestroy()

DESCRIPTION

Destroys the *RvSdpTimeZoneAdjust* field of an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgTZADestroy(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgTZAIsUsed()

DESCRIPTION

Indicates whether or not the *RvSdpTimeZoneAdjust* field ('z=') of an *RvSdpMsg* is set.

SYNTAX

```
RvBool rvSdpMsgTZAIsUsed(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns RV_TRUE if the *RvSdpTimeZoneAdjust* objects are set, or RV_FALSE otherwise.

rvSdpMsgIsBadSyntaxZoneAdjustment()

DESCRIPTION

Indicates whether or not the time *RvSdpTimeZoneAdjust* of an *RvSdpMsg* is proprietary-formatted.

SYNTAX

```
RvBool rvSdpMsgIsBadSyntaxZoneAdjustment(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpMsgTimeAddZoneAdjustment()

DESCRIPTION

Adds a new *RvSdpTimeZoneAdjust* to the list specified in the 'z=' line.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpMsgTimeAddZoneAdjustment(
    RvSdpMsg*        msg,
    RvUint32         time,
    RvInt32          adjust_time,
    RvSdpTimeUnit    units);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

time

The time at which the adjustment is applied.

adjust_time

The time shift length.

units

The units of the time shift.

RETURN VALUES

Returns a pointer to the newly created *RvSdpTimeZoneAdjust* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentZoneAdjustment()

DESCRIPTION

Removes (and destructs) an *RvSdpTimeZoneAdjust* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentZoneAdjustment(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstZoneAdjustment() or rvSdpMsgGetNextZoneAdjustment() function.

RETURN VALUES

None.

rvSdpMsgRemoveTimeZoneAdjust()

DESCRIPTION

Removes (and destructs) an *RvSdpTimeZoneAdjust* by index.

SYNTAX

```
void rvSdpMsgRemoveTimeZoneAdjust(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfZoneAdjustments().

RETURN VALUES

None.

rvSdpMsgClearZoneAdjustment()

DESCRIPTION

Removes (and destructs) all *RvSdpTimeZoneAdjust* objects set in an RvSdpMsg.

SYNTAX

```
void rvSdpMsgClearZoneAdjustment(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddAttr()

DESCRIPTION

Adds a new generic *RvSdpAttribute* at the session level.

SYNTAX

```
RvSdpAttribute* rvSdpMsgAddAttr(
    RvSdpMsg*       msg,
    const char*     name,
    const char*     value);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

name

The generic *RvSdpAttribute* name.

value

The generic *RvSdpAttribute* value.

RETURN VALUES

Returns a pointer to the newly created *RvSdpAttribute* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentAttribute()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentAttribute(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstAttribute() or rvSdpMsgGetNextAttribute() function.

RETURN VALUES

None.

rvSdpMsgRemoveAttribute()

DESCRIPTION

Removes (and destructs) a generic *RvSdpAttribute* by index.

SYNTAX

```
void rvSdpMsgRemoveAttribute(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfAttr().

RETURN VALUES

None.

rvSdpMsgClearAttr()

DESCRIPTION

Removes (and destructs) all generic *RvSdpAttribute* objects set in an *RvSdpMsg*. The special *RvSdpAttribute* will not be removed. Use rvSdpMsgClearAttr2() to remove all (generic and special) *RvSdpAttribute* objects.

SYNTAX

```
void rvSdpMsgClearAttr(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgRemoveCurrentAttribute2()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentAttribute2(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rvSdpMsgGetFirstAttribute2() or rvSdpMsgGetNextAttribute2() function.

RETURN VALUES

None.

rvSdpMsgRemoveAttribute2()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* by index.

SYNTAX

```
void rvSdpMsgRemoveAttribute2(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfAttr2().

RETURN VALUES

None.

rvSdpMsgClearAttr2()

DESCRIPTION

Removes (and destructs) all (generic and special) *RvSdpAttribute* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearAttr2(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddRtpMap()

DESCRIPTION

Adds a new *RvSdpRtpMap* to the session-level *RvSdpRtpMap* list.

SYNTAX

```
RvSdpRtpMap* rvSdpMsgAddRtpMap(
    RvSdpMsg*       msg,
    int             payload,
    const char*     encoding_name,
    int             rate);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

payload

An RTP dynamic payload number.

encoding_name

The name of the codec.

rate

The clock rate.

RETURN VALUES

Returns a pointer to the newly created *RvSdpRtpMap* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxRtpMap()

DESCRIPTION

Adds a new, proprietary-formatted *RvSdpRtpMap* at the session level.

SYNTAX

```
RvSdpRtpMap* rvSdpMsgAddBadSyntaxRtpMap(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of the *RvSdpRtpMap* special attribute.

RETURN VALUES

Returns a pointer to the newly created *RvSdpRtpMap* if the function succeeds, or NULL pointer if the function fails.

rvSdpMsgRemoveCurrentRtpMap()

DESCRIPTION

Removes (and destructs) an *RvSdpRtpMap* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentRtpMap(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstRtpMap() or rvSdpMsgGetNextRtpMap() function.

RETURN VALUES

None.

rvSdpMsgRemoveRtpMap()

DESCRIPTION

Removes (and destructs) an *RvSdpRtpMap* special attribute by index.

SYNTAX

```
void rvSdpMsgRemoveRtpMap(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfRtpMaps().

RETURN VALUES

None.

RvSdpMsgClearRtpMap()

DESCRIPTION

Removes (and destructs) all *RvSdpRtpMap* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearRtpMap(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddKeyMgmt()

DESCRIPTION

Adds a new *RvSdpKeyMgmtAttr* special attribute at the session level.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMsgAddKeyMgmt(
    RvSdpMsg*       msg,
    const char*     prtclId,
    const char*     keyData);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

prtclId

The protocol ID.

keyData

The encryption key data.

RETURN VALUES

Returns a pointer to the newly created *RvSdpKeyMgmtAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxKeyMgmt()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpKeyMgmtAttr* at the session level.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMsgAddBadSyntaxKeyMgmt(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of the *RvSdpKeyMgmtAttr* special attribute field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpKeyMgmtAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentKeyMgmt()

DESCRIPTION

Removes (and destructs) an *RvSdpKeyMgmt* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentKeyMgmt(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* set or modified by a previous, successful call to the rvSdpMsgGetFirstKeyMgmt() or rvSdpMsgGetNextKeyMgmt() function.

RETURN VALUES

None.

rvSdpMsgRemoveKeyMgmt()

DESCRIPTION

Removes (and destructs) an *RvSdpKeyMgmtAttr* special attribute by index.

SYNTAX

```
void rvSdpMsgRemoveKeyMgmt(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfKeyMgmt().

RETURN VALUES

None.

rvSdpMsgClearKeyMgmt()

DESCRIPTION

Removes (and destructs) all *RvSdpKeyMgmtAttr* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearKeyMgmt(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddCrypto()

DESCRIPTION

Adds a new *RvSdpCryptoAttr* special attribute at the session level.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMsgAddCrypto(
    RvSdpMsg*       msg,
    RvUint          tag,
    const char*     suite);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

tag

The *RvSdpCryptoAttr* tag number.

suite

The *RvSdpCryptoAttr* suite value.

RETURN VALUES

Returns a pointer to the newly created *RvSdpCryptoAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgAddBadSyntaxCrypto()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpCryptoAttr* at the session level.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMsgAddBadSyntaxCrypto(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of the *RvSdpCryptoAttr* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpCryptoAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentCrypto()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentCrypto(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstCrypto() or rvSdpMsgGetNextCrypto() function.

RETURN VALUES

None.

rvSdpMsgRemoveCrypto()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* by index.

SYNTAX

```
void rvSdpMsgRemoveCrypto(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfCrypto().

RETURN VALUES

None.

rvSdpMsgClearCrypto()

DESCRIPTION

Removes (and destructs) all *RvSdpCryptoAttr* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearCrypto(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddMediaDescr()

DESCRIPTION

Adds a new *RvSdpMediaDescr* to the *RvSdpMsg*.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgAddMediaDescr(
    RvSdpMsg*          msg,
    RvSdpMediaType     media_type,
    RvUint16           port,
    RvSdpProtocol      protocol);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

media_type

The type of media (audio, video or data).

port

The port number.

protocol

The protocol used to transport the media, such as RTP/RTCP.

RETURN VALUES

Returns a pointer to the newly created *RvSdpMediaDescr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgInsertMediaDescr()

DESCRIPTION

Adds a new *RvSdpMediaDescr* to the *RvSdpMsg* as a copy of *descr.*

SYNTAX

```
RvSdpStatus rvSdpMsgInsertMediaDescr(
    RvSdpMsg*           msg,
    RvSdpMediaDescr*    descr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where new *RvSdpMediaDescr* objects will be added.

descr

The new *RvSdpMediaDescr* will be copied from this *RvSdpMediaDescr.*

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgAddBadSyntaxMediaDescr()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpMediaDescr* at the session level.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgAddBadSyntaxMediaDescr(
    RvSdpMsg*     msg,
    const char    *badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary value of the *RvSdpMediaDescr* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpMediaDescr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentMediaDescr()

DESCRIPTION

Removes (and destructs) an *RvSdpMediaDescr* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentMediaDescr(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstMediaDescr() or rvSdpMsgGetNextMediaDescr() function.

RETURN VALUES

None.

rvSdpMsgRemoveMediaDescr()

DESCRIPTION

Removes (and destructs) an *RvSdpMediaDescr* by index.

SYNTAX

```
void rvSdpMsgRemoveMediaDescr(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfMediaDescr().

RETURN VALUES

None.

rvSdpMsgClearMediaDescr()

DESCRIPTION

Removes (and destructs) all *RvSdpMediaDescr* objects set in an *RvSdpMsg*.

SYNTAX

```
void rvSdpMsgClearMediaDescr(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

rvSdpMsgAddOther()

DESCRIPTION

Adds a new *RvSdpOther* to the session-level list of *RvSdpOther* objects.

SYNTAX

```
RvSdpOther* rvSdpMsgAddOther(
    RvSdpMsg*     msg,
    const char    tag,
    const char    *value);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

tag

The tag letter of the line.

value

The proprietary text of the line.

RETURN VALUES

Returns a pointer to the newly created *RvSdpOther* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMsgRemoveCurrentOther()

DESCRIPTION

Removes (and destructs) an *RvSdpOther* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgRemoveCurrentOther(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstOther() or rvSdpMsgGetNextOther() function.

RETURN VALUES

None.

rvSdpMsgRemoveOther()

DESCRIPTION

Removes (and destructs) an *RvSdpOther* by index.

SYNTAX

```
void rvSdpMsgRemoveOther(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfOther().

RETURN VALUES

None.

rvSdpMsgClearOther()

DESCRIPTION

Removes (and destructs) all *RvSdpOther* objects set in an RvSdpMsg.

SYNTAX

```
void rvSdpMsgClearOther(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpMsgGetVersion()
- rvSdpMsgSetVersionN()
- rvSdpMsgGetOrigin()
- rvSdpMsgSetOrigin()
- rvSdpMsgSetBadSyntaxOrigin()
- rvSdpMsgGetSessionName()
- rvSdpMsgSetSessionName()
- rvSdpMsgGetSessionInformation()
- rvSdpMsgSetSessionInformation()
- rvSdpMsgGetURI()
- rvSdpMsgSetURI()
- rvSdpGetBadSyntaxUri()
- rvSdpMsgSetBadSyntaxURI()
- rvSdpMsgGetNumOfEmail()
- rvSdpMsgGetFirstEmail()
- rvSdpMsgGetNextEmail()
- rvSdpMsgGetEmail()
- rvSdpMsgGetNumOfPhones()
- rvSdpMsgGetFirstPhone()
- rvSdpMsgGetNextPhone()
- rvSdpMsgGetPhone()
- rvSdpMsgGetNumOfConnections()
- rvSdpMsgGetFirstConnection()
- rvSdpMsgGetNextConnection()
- rvSdpMsgGetConnection()
- rvSdpMsgGetConnectionByIndex()
- rvSdpMsgSetConnection()
- rvSdpMsgSetBadSyntaxConnection()
- rvSdpMsgGetNumOfBandwidth()
- rvSdpMsgGetFirstBandwidth()
- rvSdpMsgGetNextBandwidth()
- rvSdpMsgGetBandwidth()

- rvSdpMsgGetBandwidthByIndex()
- rvSdpMsgSetBandwidth()
- rvSdpMsgSetBadSyntaxBandwidth()
- rvSdpMsgGetNumOfSessionTime()
- rvSdpMsgGetFirstSessionTime()
- rvSdpMsgGetNextSessionTime()
- rvSdpMsgGetSessionTime()
- rvSdpMsgGetBadSyntaxZoneAdjustment()
- rvSdpMsgSetBadSyntaxZoneAdjustment()
- rvSdpMsgGetNumOfZoneAdjustments()
- rvSdpMsgGetFirstZoneAdjustment()
- rvSdpMsgGetNextZoneAdjustment()
- rvSdpMsgGetZoneAdjustment()
- rvSdpMsgGetKey()
- rvSdpMsgSetKey()
- rvSdpMsgSetBadSyntaxKey()
- rvSdpMsgGetNumOfAttr()
- rvSdpMsgGetFirstAttribute()
- rvSdpMsgGetNextAttribute()
- rvSdpMsgGetAttribute()
- rvSdpMsgGetNumOfAttr2()
- rvSdpMsgGetFirstAttribute2()
- rvSdpMsgGetNextAttribute2()
- rvSdpMsgGetAttribute2()
- rvSdpMsgGetNumOfRtpMaps()
- rvSdpMsgGetFirstRtpMap()
- rvSdpMsgGetNextRtpMap()
- rvSdpMsgGetRtpMap()
- rvSdpMsgGetConnectionMode()
- rvSdpMsgSetConnectionMode()
- rvSdpMsgGetNumOfKeyMgmt()
- rvSdpMsgGetFirstKeyMgmt()
- rvSdpMsgGetNextKeyMgmt()

- rvSdpMsgGetKeyMgmt()
- rvSdpMsgGetNumOfCrypto()
- rvSdpMsgGetFirstCrypto()
- rvSdpMsgGetNextCrypto()
- rvSdpMsgGetCrypto()
- rvSdpMsgGetNumOfMediaDescr()
- rvSdpMsgGetFirstMediaDescr()
- rvSdpMsgGetNextMediaDescr()
- rvSdpMsgGetMediaDescr()
- rvSdpMsgGetNumOfOther()
- rvSdpMsgGetFirstOther()
- rvSdpMsgGetNextOther()
- rvSdpMsgGetOther()
- rvSdpMsgGetNumOfBadSyntax2()
- rvSdpMsgGetNumOfBadSyntax()
- rvSdpMsgGetFirstBadSyntax()
- rvSdpMsgGetNextBadSyntax()
- rvSdpMsgGetBadSyntax()

rvSdpMsgGetVersion()

DESCRIPTION

Gets the version field value of an *RvSdpMsg*.

SYNTAX

```
const char* rvSdpMsgGetVersion(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the text version.

rvSdpMsgSetVersionN()

DESCRIPTION

Sets the version field of the *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgSetVersionN(
    RvSdpMsg*       msg,
    const char*     version);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

version

The new version value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetOrigin()

DESCRIPTION

Gets a pointer to the origin field.

SYNTAX

```
RvSdpOrigin* rvSdpMsgGetOrigin(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the origin field, or NULL if the origin field is not set in the message.

rvSdpMsgSetOrigin()

DESCRIPTION

Sets the SDP origin field.

SYNTAX

```
RvSdpStatus rvSdpMsgSetOrigin(
    RvSdpMsg*        msg,
    const char*      username,
    const char*      session_id,
    const char*      version,
    RvSdpNetType     nettype,
    RvSdpAddrType    addrtype,
    const char*      address);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

username

The user name.

session_id

The session ID.

version

The version.

nettype

The network type.

addrtype

The address type.

address

The address, depending on the network type. For example, an IP address for an IP network, and so on.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgSetBadSyntaxOrigin()

DESCRIPTION

Sets the SDP origin field with a proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxOrigin(
    RvSdpMsg*       msg,
    const char*     origin);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

origin

The proprietary-formatted origin to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetSessionName()

DESCRIPTION

Gets the session name field value of an *RvSdpMsg*.

SYNTAX

```
const char* rvSdpMsgGetSessionName(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the session name text.

rvSdpMsgSetSessionName()

DESCRIPTION

Sets the session name field of the *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgSetSessionName(
    RvSdpMsg*      msg,
    const char*    session_name);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

session_name

The new session name value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetSessionInformation()

DESCRIPTION

Gets the session information field value of an *RvSdpMsg*.

SYNTAX

```
const char* rvSdpMsgGetSessionInformation(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the session information text.

rvSdpMsgSetSessionInformation()

DESCRIPTION

Sets the information field of the *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgSetSessionInformation(
    RvSdpMsg*       msg,
    const char*     info);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

info

The new information value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetURI()

DESCRIPTION

Gets the URI field value of an *RvSdpMsg*.

SYNTAX

```
const char* rvSdpMsgGetURI(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the URI text.

rvSdpMsgSetURI()

DESCRIPTION

Sets the URI field of the *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgSetURI(
    RvSdpMsg*      msg,
    const char*    uri);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

uri

The new URI value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpGetBadSyntaxUri()

DESCRIPTION

Gets a proprietary formatted URI value, or an empty string ("") if the value is legal or is not set.

SYNTAX

```
const char* rvSdpMsgGetBadSyntaxUri(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

The bad syntax value.

rvSdpMsgSetBadSyntaxURI()

DESCRIPTION

Sets the SDP URI field with proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxURI(
    RvSdpMsg*       msg,
    const char*     uri);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

uri

The proprietary-formatted URI to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfEmail()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpEmail* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfEmail(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of elements in the session-level list of *RvSdpEmail* objects.

rvSdpMsgGetFirstEmail()

DESCRIPTION

Returns the first *RvSdpEmail* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpEmail* rvSdpMsgGetFirstEmail(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further calls to rvSdpMsgGetNextEmail().

RETURN VALUES

Returns a pointer to the *RvSdpEmail*, or a NULL pointer if there are no *RvSdpEmail* objects defined in the *RvSdpMsg*.

rvSdpMsgGetNextEmail()

DESCRIPTION

Returns the next *RvSdpEmail* defined in the *RvSdpMsg*. The next *RvSdpEmail* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpEmail* rvSdpMsgGetNextEmail(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstEmail() function.

RETURN VALUES

Returns a pointer to the *RvSdpEmail*, or a NULL pointer if there are no more emails defined in the *RvSdpMsg*.

rvSdpMsgGetEmail()

DESCRIPTION

Gets an *RvSdpEmail* by index.

SYNTAX

```
RvSdpEmail* rvSdpMsgGetEmail(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfEmail().

RETURN VALUES

Returns the requested *RvSdpEmail*.

rvSdpMsgGetNumOfPhones()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpPhone* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfPhones(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the phone list of the *RvSdpMsg*.

rvSdpMsgGetFirstPhone()

DESCRIPTION

Returns the first *RvSdpPhone* defined in an *RvSdpMsg*. This function also sets the list *RvSdpListIter* for further use.

SYNTAX

```
RvSdpPhone* rvSdpMsgGetFirstPhone(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further calls to rvSdpMsgGetNextPhone()

RETURN VALUES

Returns a pointer to the *RvSdpPhone*, or a NULL pointer if there are no phones defined in the *RvSdpMsg*.

rvSdpMsgGetNextPhone()

DESCRIPTION

Returns the next *RvSdpPhone* defined in an *RvSdpMsg*. The next *RvSdpPhone* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpPhone* rvSdpMsgGetNextPhone(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous successful call to the rvSdpMsgGetFirstPhone() or rvSdpMsgGetNextPhone() function.

RETURN VALUES

Returns a pointer to the *RvSdpPhone*, or a NULL pointer if there are no more phones defined in the *RvSdpMsg*.

rvSdpMsgGetPhone()

DESCRIPTION

Gets an *RvSdpPhone* by index.

SYNTAX

```
RvSdpPhone* rvSdpMsgGetPhone(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfPhones().

RETURN VALUES

Returns the requested *RvSdpPhone*.

rvSdpMsgGetNumOfConnections()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpConnection* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfConnections(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the connections list of an *RvSdpMsg*.

rvSdpMsgGetFirstConnection()

DESCRIPTION

Returns the first *RvSdpConnection* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpConnection* rvSdpMsgGetFirstConnection(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further calls to rvSdpMsgGetNextConnection() calls.

RETURN VALUES

Returns a pointer to the *RvSdpConnection*, or a NULL pointer if there are no connections defined in the *RvSdpMsg*.

rvSdpMsgGetNextConnection()

DESCRIPTION

Returns the next *RvSdpConnection* defined in the *RvSdpMsg*. The next *RvSdpConnection* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpConnection* rvSdpMsgGetNextConnection(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstConnection() or rvSdpMsgGetNextConnection() function.

RETURN VALUES

Returns a pointer to the *RvSdpConnection*, or a NULL pointer if there are no more connections defined in the *RvSdpMsg*.

rvSdpMsgGetConnection()

DESCRIPTION

Gets a pointer to the first *RvSdpConnection* set in the message.

SYNTAX

```
RvSdpConnection* rvSdpMsgGetConnection(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the connection field, or NULL if there are not connection fields set in the message.

rvSdpMsgGetConnectionByIndex()

DESCRIPTION

Gets an *RvSdpConnection* by index.

SYNTAX

```
RvSdpConnection* rvSdpMsgGetConnectionByIndex(
    const RvSdpMsg*    descr,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfConnections().

RETURN VALUES

Returns the requested *RvSdpConnection*.

rvSdpMsgSetConnection()

DESCRIPTION

Adds a new *RvSdpConnection* at the session level.

SYNTAX

```
RvSdpStatus rvSdpMsgSetConnection(
    RvSdpMsg*        msg,
    RvSdpNetType     net_type,
    RvSdpAddrType    addr_type,
    const char*      addr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

net_type

The network type.

addr_type

The address type.

addr

The address, depending on the network type. For example, an IP address for an IP network, and so on.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgSetBadSyntaxConnection()

DESCRIPTION

Sets the SDP *RvSdpConnection* field with a proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxConnection(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary formatted *RvSdpConnection* to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfBandwidth()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpBandwidth* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfBandwidth(
    const RvSdpMsg*    descr);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the *RvSdpBandwidth* list of an *RvSdpMsg*.

rvSdpMsgGetFirstBandwidth()

DESCRIPTION

Returns the first *RvSdpBandwidth* defined in an *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpBandwidth* rvSdpMsgGetFirstBandwidth(
    RvSdpMsg*         descr,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextBandwidth() calls.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth*, or a NULL pointer if there are no bandwidths defined in the *RvSdpMsg*.

rvSdpMsgGetNextBandwidth()

DESCRIPTION

Returns the next *RvSdpBandwidth* defined in the *RvSdpMsg*. The next *RvSdpBandwidth* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpBandwidth* rvSdpMsgGetNextBandwidth(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstBandwidth() or rvSdpMsgGetNextBandwidth() function.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth*, or a NULL pointer if there are no more bandwidths defined in the *RvSdpMsg*.

rvSdpMsgGetBandwidth()

DESCRIPTION

Gets a pointer to the first *RvSdpBandwidth* set in the message.

SYNTAX

```
RvSdpBandwidth* rvSdpMsgGetBandwidth(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth* field, or NULL if there are no *RvSdpBandwidth* fields set in the message.

rvSdpMsgGetBandwidthByIndex()

DESCRIPTION

Gets an *RvSdpBandwidth* by index.

SYNTAX

```
RvSdpBandwidth* rvSdpMsgGetBandwidthByIndex(
    const RvSdpMsg*    descr,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfBandwidth().

RETURN VALUES

Returns the requested *RvSdpBandwidth*.

rvSdpMsgSetBandwidth()

DESCRIPTION

Adds a new *RvSdpBandwidth* at the session level.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBandwidth(
    RvSdpMsg*       msg,
    const char*     bwtype,
    int             b);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

bwtype

The *RvSdpBandwidth* type, such as Conference Total (CT) or Application-Specific Maximum (AS).

b

The *RvSdpBandwidth* value in kilobits per second (kbps).

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgSetBadSyntaxBandwidth()

DESCRIPTION

Sets the SDP *RvSdpBandwidth* field with a proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxBandwidth(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary-formatted *RvSdpBandwidth* to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfSessionTime()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpSessionTime* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfSessionTime(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the *RvSdpSessionTime* list of the *RvSdpMsg*.

rvSdpMsgGetFirstSessionTime()

DESCRIPTION

Returns the first *RvSdpSessionTime* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpSessionTime* rvSdpMsgGetFirstSessionTime(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further calls to rvSdpMsgGetNextSessionTime().

RETURN VALUES

Returns a pointer to the *RvSdpSessionTime*, or a NULL pointer if there are no session times defined in the *RvSdpMsg*.

rvSdpMsgGetNextSessionTime()

DESCRIPTION

Returns the next *RvSdpSessionTime* defined in the *RvSdpMsg*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpSessionTime* rvSdpMsgGetNextSessionTime(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rvSdpMsgGetFirstSessionTime() or rvSdpMsgGetNextSessionTime().

RETURN VALUES

Returns a pointer to the *RvSdpSessionTime*, or a NULL pointer if there are no more session times defined in the *RvSdpMsg*.

rvSdpMsgGetSessionTime()

DESCRIPTION

Gets an *RvSdpSessionTime* by index.

SYNTAX

```
RvSdpSessionTime* rvSdpMsgGetSessionTime(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfSessionTime().

RETURN VALUES

Returns the requested *RvSdpSessionTime*.

rvSdpMsgGetBadSyntaxZoneAdjustment()

DESCRIPTION

Gets a proprietary-formatted time *RvSdpTimeZoneAdjust* of an *RvSdpMsg* or an empty string ("") if the value is either legal or not set.

SYNTAX

```
const char* rvSdpMsgGetBadSyntaxZoneAdjustment(
    RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Gets the proprietary formatted time *RvSdpTimeZoneAdjust* of an *RvSdpMsg*, or an empty string ("") if the value is either legal or not set.

rvSdpMsgSetBadSyntaxZoneAdjustment()

DESCRIPTION

Sets the SDP time *RvSdpTimeZoneAdjust* with a proprietary-formatted value.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxZoneAdjustment(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfZoneAdjustments()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpTimeZoneAdjust* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfZoneAdjustments(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the time zone adjustments list of the *RvSdpMsg*.

rvSdpMsgGetFirstZoneAdjustment()

DESCRIPTION

Returns the first *RvSdpTimeZoneAdjust* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpMsgGetFirstZoneAdjustment(
    RvSdpMsg*        msg,
    RvSdpListIter    *iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextZoneAdjustment()calls.

RETURN VALUES

Returns a pointer to the RvSdpZoneAdjustment, or a NULL pointer if there are no time zone adjustments defined in the *RvSdpMsg*.

rvSdpMsgGetNextZoneAdjustment()

DESCRIPTION

Returns the next *RvSdpTimeZoneAdjust* defined in the *RvSdpMsg*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpMsgGetNextZoneAdjustment(
    RvSdpListIter    *iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstZoneAdjustment() or rvSdpMsgGetNextZoneAdjustment() function.

RETURN VALUES

Returns a pointer to the *RvSdpTimeZoneAdjust*, or a NULL pointer if there are no more time zone adjustments defined in the *RvSdpMsg*.

rvSdpMsgGetZoneAdjustment()

DESCRIPTION

Gets an *RvSdpTimeZoneAdjust* by index.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpMsgGetZoneAdjustment(
    RvSdpMsg*    msg,
    RvSize_t     index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfZoneAdjustments().

RETURN VALUES

Returns the requested *RvSdpTimeZoneAdjust*.

rvSdpMsgGetKey()

DESCRIPTION

Gets a pointer to the key field.

SYNTAX

```
RvSdpKey* rvSdpMsgGetKey(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns a pointer to the key field, or NULL if the key field is not set in the message.

rvSdpMsgSetKey()

DESCRIPTION

Sets the SDP key field.

SYNTAX

```
RvSdpStatus rvSdpMsgSetKey(
    RvSdpMsg*           msg,
    RvSdpEncrMethod     em,
    const char*         key);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

em

The key encryption method.

key

The key value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgSetBadSyntaxKey()

DESCRIPTION

Sets the SDP key field with a proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMsgSetBadSyntaxKey(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary-formatted key to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfAttr()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpAttribute* objects (generic attributes).

Special *RvSdpAttribute* objects (*RvSdpRtpMap*, connection mode, *RvSdpKeyMgmtAttr*, *RvSdpCryptoAttr*, framerate and *fmtp*) are not counted among the *RvSdpAttribute* objects treated by this function. Use the rvSdpMsgGetNumOfAttr2() function to get the total number of special and generic attributes.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfAttr(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of generic *RvSdpAttribute* objects of the message (with the exception of the special *RvSdpAttribute* objects).

rvSdpMsgGetFirstAttribute()

DESCRIPTION

Returns the first *RvSdpAttribute* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use. Use rvSdpMsgGetFirstAttribute2() for iterating on all (generic and special) *RvSdpAttribute* objects.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetFirstAttribute(
    RvSdpMsg*          msg,
    RvSdpListIter*     iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextAttribute() calls.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if no generic attributes are defined in the *RvSdpMsg*.

rvSdpMsgGetNextAttribute()

DESCRIPTION

Returns the next *RvSdpAttribute* defined in the *RvSdpMsg*. The next *RvSdpAttribute* is defined based on the state of the *RvSdpListIter*. Use rvSdpMsgGetNextAttribute2() for iterating on all (generic and special) *RvSdpAttribute* objects.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetNextAttribute(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstAttribute() or rvSdpMsgGetNextAttribute() function.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no more generic attributes defined in the *RvSdpMsg*.

rvSdpMsgGetAttribute()

DESCRIPTION

Gets an *RvSdpAttribute* by index. Use rvSdpMsgGetAttribute2() to get the attribute of all (generic and special) *RvSdpAttribute* objects.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetAttribute(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfAttr().

RETURN VALUES

Returns the requested *RvSdpAttribute*.

rvSdpMsgGetNumOfAttr2()

DESCRIPTION

Gets the number of elements in the session-level list of *RvSdpAttribute* objects. Special *RvSdpAttribute* objects are counted in addition to the generic objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfAttr2(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the size of the *RvSdpAttribute* list of the *RvSdpMsg*.

rvSdpMsgGetFirstAttribute2()

DESCRIPTION

Returns the first *RvSdpAttribute* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetFirstAttribute2(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextAttribute2() calls.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no attributes defined in the *RvSdpMsg*.

rvSdpMsgGetNextAttribute2()

DESCRIPTION

Returns the next *RvSdpAttribute* defined in the *RvSdpMsg*. The next *RvSdpAttribute* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetNextAttribute2(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* set or modified by a previous, successful call to the rvSdpMsgGetFirstAttribute2() or rvSdpMsgGetNextAttribute2() function.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no more attributes defined in the *RvSdpMsg*.

rvSdpMsgGetAttribute2()

DESCRIPTION

Gets an *RvSdpAttribute* by index.

SYNTAX

```
RvSdpAttribute* rvSdpMsgGetAttribute2(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfAttr2().

RETURN VALUES

Returns the requested *RvSdpAttribute*.

rvSdpMsgGetNumOfRtpMaps()

DESCRIPTION

Gets the number of *RvSdpAttribute* objects of an *RvSdpRtpMap* set in the context of an *RvSdpMsg*.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfRtpMap(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpAttribute* objects of an *RvSdpRtpMap* set in the context of an *RvSdpMsg*.

rvSdpMsgGetFirstRtpMap()

DESCRIPTION

Returns the first *RvSdpRtpMap* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpRtpMap* rvSdpMsgGetFirstRtpMap(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextRtpMap() calls.

RETURN VALUES

Returns a pointer to the *RvSdpRtpMap*, or a NULL pointer if there are no RTP maps defined in the *RvSdpMsg*.

rvSdpMsgGetNextRtpMap()

DESCRIPTION

Returns the next *RvSdpRtpMap* defined in the *RvSdpMsg*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpRtpMap* rvSdpMsgGetNextRtpMap(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstRtpMap() or rvSdpMsgGetNextRtpMap() function.

RETURN VALUES

Returns a pointer to the *RvSdpRtpMap*, or a NULL pointer if there are no more RTP maps defined in the *RvSdpMsg*.

rvSdpMsgGetRtpMap()

DESCRIPTION

Gets an *RvSdpRtpMap* special attribute by index.

SYNTAX

```
RvSdpRtpMap* rvSdpMsgGetRtpMap(
    const RvSdpMsg*    vmsg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfRtpMaps().

RETURN VALUES

Returns the requested *RvSdpRtpMap* attribute.

rvSdpMsgGetConnectionMode()

DESCRIPTION

Returns the connection mode of the *RvSdpMsg* or RV_SDPCONNECTMODE_NOTSET if the corresponding special attribute is not set.

SYNTAX

```
RvSdpConnectionMode rvSdpMsgGetConnectionMode(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the connection mode.

rvSdpMsgSetConnectionMode()

DESCRIPTION

Sets or modifies the connection mode of an *RvSdpMsg*.

SYNTAX

```
RvSdpStatus rvSdpMsgSetConnectionMode(
    RvSdpMsg*              msg,
    RvSdpConnectionMode    mode);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

mode

The new value of connection mode.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetNumOfKeyMgmt()

DESCRIPTION

Gets the number of *RvSdpKeyMgmtAttr* objects set in the context of an *RvSdpMsg*.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfKeyMgmt(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

The number of *RvSdpKeyMgmtAttr* objects set in the context of an *RvSdpMsg*.

rvSdpMsgGetFirstKeyMgmt()

DESCRIPTION

Returns the first *RvSdpKeyMgmtAttr* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMsgGetFirstKeyMgmt(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextKeyMgmt() calls.

RETURN VALUES

Returns a pointer to the *RvSdpKeyMgmtAttr*, or a NULL pointer if there are no *RvSdpKeyMgmtAttr* objects defined in the *RvSdpMsg*.

rvSdpMsgGetNextKeyMgmt()

DESCRIPTION

Returns the next *RvSdpKeyMgmtAttr* special attribute defined in the *RvSdpMsg*. The next *RvSdpKeyMgmtAttr* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMsgGetNextKeyMgmt(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* set or modified by a previous, successful call to the rvSdpMsgGetFirstKeyMgmt() or rvSdpMsgGetNextKeyMgmt() function.

RETURN VALUES

Returns a pointer to the *RvSdpKeyMgmtAttr*, or a NULL pointer if there are no more *RvSdpKeyMgmtAttr* objects defined in the *RvSdpMsg*.

rvSdpMsgGetKeyMgmt()

DESCRIPTION

Gets an *RvSdpKeyMgmtAttr* by index.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMsgGetKeyMgmt(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfKeyMgmt().

RETURN VALUES

Returns the requested *RvSdpKeyMgmtAttr*.

rvSdpMsgGetNumOfCrypto()

DESCRIPTION

Gets the number of *RvSdpCryptoAttr* objects set in the context of an *RvSdpMsg*.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfCrypto(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpCryptoAttr* objects set in the context of an *RvSdpMsg*.

rvSdpMsgGetFirstCrypto()

DESCRIPTION

Returns the first *RvSdpCryptoAttr* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMsgGetFirstCrypto(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextCrypto() calls.

RETURN VALUES

A pointer to the *RvSdpCryptoAttr*, or a NULL pointer if there are no crypto objects defined in the *RvSdpMsg*.

rvSdpMsgGetNextCrypto()

DESCRIPTION

Returns the next *RvSdpCryptoAttr* defined in an *RvSdpMsg*. The next *RvSdpCryptoAttr* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMsgGetNextCrypto(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstCrypto() or rvSdpMsgGetNextCrypto() function.

RETURN VALUES

A pointer to the *RvSdpCryptoAttr*, or a NULL pointer if there are no more *RvSdpCryptoAttr* objects defined in the *RvSdpMsg*.

rvSdpMsgGetCrypto()

DESCRIPTION

Gets an *RvSdpCryptoAttr* by index.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMsgGetCrypto(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfCrypto().

RETURN VALUES

Returns the requested *RvSdpCryptoAttr*.

rvSdpMsgGetNumOfMediaDescr()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* objects set in the context of an *RvSdpMsg*.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfMediaDescr(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpMediaDescr* objects set in the *RvSdpMsg*.

rvSdpMsgGetFirstMediaDescr()

DESCRIPTION

Returns the first *RvSdpMediaDescr* defined in an *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgGetFirstMediaDescr(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextMediaDescr() calls.

RETURN VALUES

A pointer to the *RvSdpMediaDescr*, or a NULL pointer if there are no *RvSdpMediaDescr* objects defined in the *RvSdpMsg*.

rvSdpMsgGetNextMediaDescr()

DESCRIPTION

Returns the next *RvSdpMediaDescr* defined in the *RvSdpMsg*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgGetNextMediaDescr(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstMediaDescr() or rvSdpMsgGetNextMediaDescr() function.

RETURN VALUES

A pointer to the *RvSdpMediaDescr*, or a NULL pointer if there are no more RvSdpMediaDescr objects defined in the *RvSdpMsg*.

rvSdpMsgGetMediaDescr()

DESCRIPTION

Gets an *RvSdpMediaDescr* by index.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgGetMediaDescr(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfMediaDescr().

RETURN VALUES

Returns the requested *RvSdpMediaDescr.*

rvSdpMsgGetNumOfOther()

DESCRIPTION

Gets the number of *RvSdpOther* objects set in the context of an *RvSdpMsg*.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfOther(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpOther* objects set in the *RvSdpMsg*.

rvSdpMsgGetFirstOther()

DESCRIPTION

Returns the first *RvSdpOther* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpOther* rvSdpMsgGetFirstOther(
    RvSdpMsg*         msg,
    RvSdpListIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextOther() calls.

RETURN VALUES

A pointer to the *RvSdpOther*, or a NULL pointer if there are no *RvSdpOther* objects defined in the *RvSdpMsg*.

rvSdpMsgGetNextOther()

DESCRIPTION

Returns the next *RvSdpOther* defined in the *RvSdpMsg*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpOther* rvSdpMsgGetNextOther(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstOther() or rvSdpMsgGetNextOther() function.

RETURN VALUES

A pointer to the *RvSdpOther*, or a NULL pointer if there are no more *RvSdpOther* objects defined in the *RvSdpMsg*.

rvSdpMsgGetOther()

DESCRIPTION

Gets an *RvSdpOther* by index.

SYNTAX

```
RvSdpOther* rvSdpMsgGetOther(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfOther().

RETURN VALUES

Returns the requested *RvSdpOther*.

rvSdpMsgGetNumOfBadSyntax2()

DESCRIPTION

Gets the number of proprietary formatted elements (of all types) in an *RvSdpMsg*, including bad syntax elements set in the context of *RvSdpMediaDescr* objects.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfBadSyntax2(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpBadSyntax* of the RvSdpMsg.

rvSdpMsgGetNumOfBadSyntax()

DESCRIPTION

Gets the number of proprietary formatted elements (of all types) in the RvSdpMsg, excluding bad syntax elements set in the context of *RvSdpMediaDescr* objects. If some of the *RvSdpMediaDescr* objects are proprietary-formatted, they are counted, but none of the *RvSdpBadSyntax* objects owned by this *RvSdpMediaDescr* are counted.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfBadSyntax(
    const RvSdpMsg*    msg);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

RETURN VALUES

Returns the number of *RvSdpBadSyntax* objects of the RvSdpMsg.

rvSdpMsgGetFirstBadSyntax()

DESCRIPTION

Returns the first *RvSdpBadSyntax* defined in the RvSdpMsg. This function also sets an *RvSdpListIter* for further use. The *RvSdpBadSyntax* objects owned by *RvSdpMediaDescr* objects are not treated by this function.

SYNTAX

```
RvSdpBadSyntax* rvSdpMsgGetFirstBadSyntax(
    RvSdpMsg*            msg,
    RvSdpLineObjIter*    iter);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextBadSyntax() calls.

RETURN VALUES

Returns a pointer to the *RvSdpBadSyntax*, or a NULL pointer if no *RvSdpBadSyntax* objects are defined for the RvSdpMsg.

rvSdpMsgGetNextBadSyntax()

DESCRIPTION

Gets the next *RvSdpBadSyntax* defined in an RvSdpMsg. The next *RvSdpBadSyntax* is defined based on the state of the *RvSdpListIter*. The bad *RvSdpBadSyntax* objects owned by *RvSdpMediaDescr* objects are not treated by the function.

SYNTAX

```
RvSdpBadSyntax* rvSdpMsgGetNextBadSyntax(
    RvSdpLineObjIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstBadSyntax() or rvSdpMsgGetNextBadSyntax() function.

RETURN VALUES

Returns a pointer to the *RvSdpBadSyntax*, or a NULL pointer if there are no more *RvSdpBadSyntax* objects defined in the RvSdpMsg.

rvSdpMsgGetBadSyntax()

DESCRIPTION

Gets an *RvSdpBadSyntax* by index. The *RvSdpBadSyntax* objects owned by the *RvSdpMediaDescr* objects are not treated by the function.

SYNTAX

```
RvSdpBadSyntax* rvSdpMsgGetBadSyntax(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMsgGetNumOfBadSyntax().

RETURN VALUES

Returns the requested *RvSdpBadSyntax*.

5

MEDIA DESCRIPTOR FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the SDP Media Descriptors API functions.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMediaDescrConstruct() **Obsolete**
- rvSdpMediaDescrConstructA() **Obsolete**
- rvSdpMediaDescrConstructCopy() **Obsolete**
- rvSdpMediaDescrConstructCopyA() **Obsolete**
- rvSdpBadSyntaxMediaDescrConstruct() **Obsolete**
- rvSdpBadSyntaxMediaDescrConstructA() **Obsolete**
- rvSdpMediaDescrIsBadSyntax()
- rvSdpMediaDescrDestruct()
- rvSdpMediaDescrCopy() **Obsolete**
- rvSdpMediaDescrAddFormatN()
- rvSdpMediaDescrAddFormat()
- rvSdpMediaDescrRemoveFormat()
- rvSdpMediaDescrClearFormat()
- rvSdpMediaDescrAddPayloadNumber()
- rvSdpMediaDescrRemovePayloadNumber()
- rvSdpMediaDescrClearPayloads()
- rvSdpMediaDescrDestroyInformation()
- rvSdpMediaDescrAddConnection()
- rvSdpMediaDescrRemoveCurrentConnection()
- rvSdpMediaDescrRemoveConnection()
- rvSdpMediaDescrClearConnection()
- rvSdpMediaDescrAddBandwidth()
- rvSdpMediaDescrRemoveCurrentBandwidth()
- rvSdpMediaDescrRemoveBandwidth()
- rvSdpMediaDescrClearBandwidth()
- rvSdpMediaDescrAddAttr()
- rvSdpMediaDescrRemoveCurrentAttribute()
- rvSdpMediaDescrRemoveAttribute()
- rvSdpMediaDescrClearAttr()
- rvSdpMediaDescrRemoveCurrentAttribute2()
- rvSdpMediaDescrRemoveAttribute2()
- rvSdpMediaDescrClearAttr2()

- rvSdpMediaDescrAddRtpMap()
- rvSdpMediaDescrAddBadSyntaxRtpMap()
- rvSdpMediaDescrRemoveCurrentRtpMap()
- rvSdpMediaDescrRemoveRtpMap()
- rvSdpMediaDescrClearRtpMap()
- rvSdpMediaDescrAddKeyMgmt()
- rvSdpMediaDescrAddBadSyntaxKeyMgmt()
- rvSdpMediaDescrRemoveCurrentKeyMgmt()
- rvSdpMediaDescrRemoveKeyMgmt()
- rvSdpMediaDescrClearKeyMgmt()
- rvSdpMediaDescrAddCrypto()
- rvSdpMediaDescrAddBadSyntaxCrypto()
- rvSdpMediaDescrRemoveCurrentCrypto()
- rvSdpMediaDescrRemoveCrypto()
- rvSdpMediaDescrClearCrypto()
- rvSdpMediaDescrDestroyFrameRate()
- rvSdpMediaDescrAddFmtp()
- rvSdpMediaDescrRemoveCurrentFmtp()
- rvSdpMediaDescrRemoveFmtp()
- rvSdpMediaDescrClearFmtp()
- rvSdpMediaDescrAddOther()
- rvSdpMediaDescrRemoveCurrentOther()
- rvSdpMediaDescrRemoveOther()
- rvSdpMediaDescrClearOther()

rvSdpMediaDescrConstruct()

DESCRIPTION

Constructs an *RvSdpMediaDescr* using the default *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrConstruct(
    RvSdpMediaDescr*    descr,
    RvSdpMediaType      mediaType,
    RvUint32            port,
    RvSdpProtocol       protocol);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr.*

mediaType

The type of media.

port

The port of the media.

protocol

The protocol of the media.

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpMediaDescrConstructA()

DESCRIPTION

Constructs the *RvSdpMediaDescr*.

This function is obsolete. Please use rvSdpMsgAddMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrConstructA(
    RvSdpMediaDescr*    descr,
    RvSdpMediaType      mediaType,
    RvUint32            port,
    RvSdpProtocol       protocol,
    RvAlloc*            a);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr*.

mediaType

The type of media.

port

The port of the media.

protocol

The protocol of the media.

badSyn

The proprietary-formatted media field, or NULL if standard media is constructed.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpMediaDescrConstructCopy()

DESCRIPTION

Constructs an *RvSdpMediaDescr* and copies the values from a source *RvSdpMediaDescr*.

This function is obsolete. Please use rvSdpMsgInsertMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrConstructCopy(
    RvSdpMediaDescr*          dest,
    const RvSdpMediaDescr*    src);
```

PARAMETERS

dest

A pointer to an *RvSdpMediaDescr* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpMediaDescr*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpMediaDescrConstructCopyA()

DESCRIPTION

Constructs an *RvSdpMediaDescr* and copies the values from a source *RvSdpMediaDescr*.

This function is obsolete. Please use rvSdpMsgInsertMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrConstructCopyA(
    RvSdpMediaDescr*          dest,
    const RvSdpMediaDescr*    src,
    RvAlloc*                  a);
```

PARAMETERS

dest

A pointer to the *RvSdpMediaDescr* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpMediaDescr*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpBadSyntaxMediaDescrConstruct()

DESCRIPTION

Constructs an *RvSdpMediaDescr* with proprietary format using the default *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpBadSyntaxMediaDescrConstruct(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr.*

badSyn

The proprietary format of the *RvSdpMediaDescr.*

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpBadSyntaxMediaDescrConstructA()

DESCRIPTION

Constructs an *RvSdpMediaDescr* with proprietary format using provided *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxMediaDescr() instead.

SYNTAX

```
RvSdpMediaDescr* rvSdpBadSyntaxMediaDescrConstructA(
    RvSdpMediaDescr*    descr,
    const char*         badSyn,
    RvAlloc*            a);
```

PARAMETERS

descr

A pointer to valid *RvSdpMediaDescr*.

badSyn

The proprietary format of the *RvSdpMediaDescr*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpMediaDescrIsBadSyntax()

DESCRIPTION

Tests whether the *RvSdpMediaDescr* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpMediaDescrIsBadSyntax(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpMediaDescrDestruct()

DESCRIPTION

Destructs an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrDestruct(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrCopy()

DESCRIPTION

Copies the values from a source *RvSdpMediaDescr* to a destination *RvSdpMediaDescr*.

This function is obsolete.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrCopy(
    RvSdpMediaDescr*          dest,
    const RvSdpMediaDescr*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpMediaDescr*. This parameter must point to a constructed *RvSdpMediaDescr*.

src

The source *RvSdpMediaDescr*.

RETURN VALUES

Returns a pointer to the destination *RvSdpMediaDescr*, or NULL if the function fails.

rvSdpMediaDescrAddFormatN()

DESCRIPTION

Adds additional codec format to the *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrAddFormatN(
    RvSdpMediaDescr*    descr,
    const char*         fmt,
    int                 len);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

fmt

The name of the format.

len

The length of *fmt*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrAddFormat()

DESCRIPTION

Adds an additional codec format to the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrAddFormat(
    RvSdpMediaDescr*    descr,
    const char*         fmt);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

fmt

The name of the format.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrRemoveFormat()

DESCRIPTION

Removes (and de-allocates) the codec format name by index in the context of the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveFormat(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfConnections().

RETURN VALUES

None.

rvSdpMediaDescrClearFormat()

DESCRIPTION

Removes (and destructs) all codec formats set in the *RvSdpMediaDescr.*

SYNTAX

```
void rvSdpMediaDescrClearFormat(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrAddPayloadNumber()

DESCRIPTION

Adds an additional payload number to the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrAddPayloadNumber(
    RvSdpMediaDescr*    descr,
    int                 payload);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

payload

The payload to be added.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrRemovePayloadNumber()

DESCRIPTION

Removes (and destructs) an codec payload number by index in the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemovePayloadNumber(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfPayloads().

RETURN VALUES

None.

rvSdpMediaDescrClearPayloads()

DESCRIPTION

Removes (and destructs) all codec payload numbers set in the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearPayloads(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrDestroyInformation()

DESCRIPTION

Destroys the information field of the *RvSdpMediaDescr.*

SYNTAX

```
void rvSdpMediaDescrDestroyInformation(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrAddConnection()

DESCRIPTION

Adds a new *RvSdpConnection* to the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpConnection* rvSdpMediaDescrAddConnection(
    RvSdpMediaDescr*    descr,
    RvSdpNetType        net_type,
    RvSdpAddrType       addr_type,
    const char*         addr);
```

PARAMETERS

descr

A pointer to the valid *RvSdpMediaDescr.*

net_type

The network type.

addr_type

The address type.

addr

The address, depending on the network type. For example, an IP address for an IP network, and so on.

RETURN VALUES

Returns a pointer to the newly created *RvSdpConnection* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentConnection()

DESCRIPTION

Removes (and destructs) an *RvSdpConnection* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentConnection(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstConnection() or rvSdpMediaDescrGetNextConnection() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveConnection()

DESCRIPTION

Removes (and destructs) an *RvSdpConnection* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveConnection(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfConnections().

RETURN VALUES

None.

rvSdpMediaDescrClearConnection()

DESCRIPTION

Removes (and destructs) all *RvSdpConnection* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearConnection(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddBandwidth()

DESCRIPTION

Adds a new *RvSdpBandwidth* at the media-descriptor level.

SYNTAX

```
RvSdpBandwidth* rvSdpMediaDescrAddBandwidth(
    RvSdpMediaDescr*    descr,
    const char*         bwtype,
    int                 b);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

bwtype

The bandwidth type, such as Conference Total (CT) or Application-Specific Maximum (AS).

b

The *RvSdpBandwidth* value in kilobits per second (kbps).

RETURN VALUES

Returns a pointer to the added *RvSdpBandwidth*, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentBandwidth()

DESCRIPTION

Removes (and destructs) an *RvSdpBandwidth* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentBandwidth(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstBandwidth() or rvSdpMediaDescrGetNextBandwidth() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveBandwidth()

DESCRIPTION

Removes (and destructs) an *RvSdpBandwidth* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveBandwidth(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfBandwidth().

RETURN VALUES

None.

rvSdpMediaDescrClearBandwidth()

DESCRIPTION

Removes (and destructs) all *RvSdpBandwidth* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearBandwidth(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddAttr()

DESCRIPTION

Adds a new generic *RvSdpAttribute* to an *RvSdpMediaDescr*.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrAddAttr(
    RvSdpMediaDescr*    descr,
    const char*         name,
    const char*         value);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

name

The name of the new generic *RvSdpAttribute*.

value

The value of the new generic *RvSdpAttribute*.

RETURN VALUES

Returns a pointer to the added *RvSdpAttribute*, or NULL if the function fails.

rvSdpMediaDescrRemoveCurrentAttribute()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* to which the *iter* parameter points in the context of an *RvSdpMediaDescr*. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentAttribute(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstAttribute() or rvSdpMediaDescrGetNextAttribute() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveAttribute()

DESCRIPTION

Removes (and destructs) a generic *RvSdpAttribute* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveAttribute(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfAttr() call.

RETURN VALUES

None.

rvSdpMediaDescrClearAttr()

DESCRIPTION

Removes (and destructs) all generic *RvSdpAttribute* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearAttr(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrRemoveCurrentAttribute2()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentAttribute2(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstAttribute2() or rvSdpMediaDescrGetNextAttribute2() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveAttribute2()

DESCRIPTION

Removes (and destructs) an *RvSdpAttribute* (generic or special) by index in the context of an *RvSdpMediaDescr.*

SYNTAX

```
void rvSdpMediaDescrRemoveAttribute2(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfAttr2() call.

RETURN VALUES

None.

rvSdpMediaDescrClearAttr2()

DESCRIPTION

Removes (and destructs) all *RvSdpAttribute* objects (generic and special) set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearAttr2(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddRtpMap()

DESCRIPTION

Adds a new *RvSdpRtpMap* to the *RvSdpRtpMap* list of an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpRtpMap* rvSdpMediaDescrAddRtpMap(
    RvSdpMediaDescr*    descr,
    int                 payload,
    const char*         encoding_name,
    int                 rate);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

payload

An RTP dynamic payload number.

encoding_name

The name of the codec.

rate

The clock rate.

RETURN VALUES

Returns a pointer to the newly created *RvSdpRtpMap* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrAddBadSyntaxRtpMap()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpRtpMap* at the media-descriptor level.

SYNTAX

```
RvSdpRtpMap* rvSdpMediaDescrAddBadSyntaxRtpMap(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

badSyn

The proprietary value of the *RvSdpRtpMap* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpRtpMap* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentRtpMap()

DESCRIPTION

Removes (and destructs) an *RvSdpRtpMap* in the context of *RvSdpMediaDescr* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentRtpMap(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstRtpMap() or rvSdpMediaDescrGetNextRtpMap() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveRtpMap()

DESCRIPTION

Removes (and destructs) an *RvSdpRtpMap* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveRtpMap(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfRtpMap().

RETURN VALUES

None.

rvSdpMediaDescrClearRtpMap()

DESCRIPTION

Removes (and destructs) all *RvSdpRtpMap* objects set in an *RvSdpMediaDescr.*

SYNTAX

```
void rvSdpMediaDescrClearRtpMap(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrAddKeyMgmt()

DESCRIPTION

Adds a new *RvSdpKeyMgmtAttr* to the RvSdpMiaDescr.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMediaDescrAddKeyMgmt(
    RvSdpMediaDescr*    descr,
    const char*         prtclId,
    const char*         keyData);
```

PARAMETERS

descr

A pointer to the RvSdpMiaDescr.

prtclId

The protocol ID.

keyData

The encryption key data.

RETURN VALUES

Returns a pointer to the newly created *RvSdpKeyMgmtAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrAddBadSyntaxKeyMgmt()

DESCRIPTION

Adds a new proprietary formatted *RvSdpKeyMgmtAttr* at the media-descriptor level.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMediaDescrAddBadSyntaxKeyMgmt(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

badSyn

The proprietary value of the *RvSdpKeyMgmtAttr* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpKeyMgmtAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentKeyMgmt()

DESCRIPTION

Removes (and destructs) an *RvSdpKeyMgmtAttr* to which the *iter* parameter points in the context of an *RvSdpMediaDescr*. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentKeyMgmt(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstKeyMgmt() or rvSdpMediaDescrGetNextKeyMgmt() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveKeyMgmt()

DESCRIPTION

Removes (and destructs) an *RvSdpKeyMgmtAttr* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveKeyMgmt(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfKeyMgmt().

RETURN VALUES

None.

rvSdpMediaDescrClearKeyMgmt()

DESCRIPTION

Removes (and destructs) all *RvSdpKeyMgmtAttr* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearKeyMgmt(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddCrypto()

DESCRIPTION

Adds a new *RvSdpCryptoAttr* to an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpCryptoAttr* rvSdpMediaDescrAddCrypto(
    RvSdpMediaDescr*    descr,
    RvUint              tag,
    const char*         suite);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

tag

The crypto attribute tag number.

suite

The crypto attribute suite value.

RETURN VALUES

Returns a pointer to the newly created *RvSdpCryptoAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrAddBadSyntaxCrypto()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpCryptoAttr* at the media- descriptor level.

SYNTAX

```
RvSdpCryptoAttr* RvSdpMediaDescrAddBadSyntaxCrypto(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

badSyn

The proprietary value of the *RvSdpCryptoAttr* field.

RETURN VALUES

Returns a pointer to the newly created *RvSdpCryptoAttr* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentCrypto()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* to which the *iter* parameter points in the context of *RvSdpMediaDescr*. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentCrypto(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstCrypto() and rvSdpMediaDescrGetNextCrypto() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveCrypto()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveCrypto(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfCrypto().

RETURN VALUES

None.

rvSdpMediaDescrClearCrypto()

DESCRIPTION

Removes (and destructs) all *RvSdpCryptoAttr* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearCrypto(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrDestroyFrameRate()

DESCRIPTION

Destroys the frame-rate special attribute of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrDestroyFrameRate(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddFmtp()

DESCRIPTION

Adds a new *fmtp* special attribute to an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrAddFmtp(
    RvSdpMediaDescr*    descr,
    const char*         val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The value of the new *fmtp* special attribute.

RETURN VALUES

Returns a pointer to the newly created *RvSdpAttribute* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentFmtp()

DESCRIPTION

Removes (and destructs) an *fmtp* special attribute to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentFmtp(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstFmtp() or rvSdpMediaDescrGetNextFmtp() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveFmtp()

DESCRIPTION

Removes (and destructs) an *fmtp* special attribute by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveFmtp(
    RvSdpMediaDescr*    descr,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfFmtp().

RETURN VALUES

None.

rvSdpMediaDescrClearFmtp()

DESCRIPTION

Removes (and destructs) all *fmtp* special attributes set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearFmtp(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

rvSdpMediaDescrAddOther()

DESCRIPTION

Adds a new *RvSdpOther* to the list of *RvSdpOther* objects of an *RvSdpMediaDescr*.

SYNTAX

```
RvSdpOther* rvSdpMediaDescrAddOther(
    RvSdpMediaDescr*    media,
    const char          tag,
    const char          *value);
```

PARAMETERS

media

A pointer to the *RvSdpMediaDescr*.

tag

The tag letter of the line.

value

The proprietary text of the line.

RETURN VALUES

Returns a pointer to the newly created *RvSdpOther* if the function succeeds, or a NULL pointer if the function fails.

rvSdpMediaDescrRemoveCurrentOther()

DESCRIPTION

Removes (and destructs) an *RvSdpOther* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMediaDescrRemoveCurrentOther(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstOther() or rvSdpMediaDescrGetNextOther() function.

RETURN VALUES

None.

rvSdpMediaDescrRemoveOther()

DESCRIPTION

Removes (and destructs) an *RvSdpOther* by index in the context of an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrRemoveOther(
    RvSdpMediaDescr*    media,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfOther().

RETURN VALUES

None.

rvSdpMediaDescrClearOther()

DESCRIPTION

Removes (and destructs) all *RvSdpOther* objects set in an *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrClearOther(
    RvSdpMediaDescr*    media);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpMediaDescrGetBadSyntaxValue()
- rvSdpMediaDescrSetBadSyntax()
- rvSdpMediaDescrGetNumOfFormats()
- rvSdpMediaDescrGetFormat()
- rvSdpMediaDescrGetNumOfPayloads()
- rvSdpMediaDescrGetPayload()
- rvSdpMediaDescrGetMediaType()
- rvSdpMediaDescrSetMediaType()
- rvSdpMediaDescrGetMediaTypeStr()
- rvSdpMediaDescrSetMediaTypeStr()
- rvSdpMediaDescrGetProtocol()
- rvSdpMediaDescrSetProtocol()
- rvSdpMediaDescrGetProtocolStr()
- rvSdpMediaDescrSetProtocolStr()
- rvSdpMediaDescrGetPort()
- rvSdpMediaDescrSetPort()
- rvSdpMediaDescrGetNumOfPorts()
- rvSdpMediaDescrSetNumOfPorts()
- rvSdpMediaDescrGetInformation()
- rvSdpMediaDescrSetInformation()
- rvSdpMediaDescrGetNumOfConnections()
- rvSdpMediaDescrGetFirstConnection()
- rvSdpMediaDescrGetNextConnection()
- rvSdpMediaDescrGetConnection()
- rvSdpMediaDescrGetConnectionByIndex()
- rvSdpMediaDescrSetConnection()
- rvSdpMediaDescrSetBadSyntaxConnection()
- rvSdpMediaDescrGetNumOfBandwidth()
- rvSdpMediaDescrGetFirstBandwidth()
- rvSdpMediaDescrGetNextBandwidth()
- rvSdpMediaDescrGetBandwidth()
- rvSdpMediaDescrGetBandwidthByIndex()

- rvSdpMediaDescrSetBandwidth()
- rvSdpMediaDescrSetBadSyntaxBandwidth()
- rvSdpMediaDescrGetKey()
- rvSdpMediaDescrSetKey()
- rvSdpMediaDescrSetBadSyntaxKey()
- rvSdpMediaDescrGetNumOfAttr()
- rvSdpMediaDescrGetFirstAttribute()
- rvSdpMediaDescrGetNextAttribute()
- rvSdpMediaDescrGetAttribute()
- rvSdpMediaDescrGetNumOfAttr2()
- rvSdpMediaDescrGetFirstAttribute2()
- rvSdpMediaDescrGetNextAttribute2()
- rvSdpMediaDescrGetAttribute2()
- rvSdpMediaDescrGetNumOfRtpMap()
- rvSdpMediaDescrGetFirstRtpMap()
- rvSdpMediaDescrGetNextRtpMap()
- rvSdpMediaDescrGetRtpMap()
- rvSdpMediaDescrGetConnectionMode()
- rvSdpMediaDescrSetConnectionMode()
- rvSdpMediaDescrGetNumOfKeyMgmt()
- rvSdpMediaDescrGetFirstKeyMgmt()
- rvSdpMediaDescrGetNextKeyMgmt()
- rvSdpMediaDescrGetKeyMgmt()
- rvSdpMediaDescrGetNumOfCrypto()
- rvSdpMediaDescrGetFirstCrypto()
- rvSdpMediaDescrGetNextCrypto()
- rvSdpMediaDescrGetCrypto()
- rvSdpMediaDescrGetFrameRate()
- rvSdpMediaDescrSetFrameRate()
- rvSdpMediaDescrGetNumOfFmtp()
- rvSdpMediaDescrGetFirstFmtp()
- rvSdpMediaDescrGetNextFmtp()
- rvSdpMediaDescrGetFmtp()

- rvSdpMediaDescrGetNumOfOther()
- rvSdpMediaDescrGetFirstOther()
- rvSdpMediaDescrGetNextOther()
- rvSdpMediaDescrGetOther()
- rvSdpMediaDescrGetNumOfBadSyntax()
- rvSdpMediaDescrGetFirstBadSyntax()
- rvSdpMediaDescrGetNextBadSyntax()
- rvSdpMediaDescrGetBadSyntax()

rvSdpMediaDescrGetBadSyntaxValue()

DESCRIPTION

Gets a proprietary-formatted *RvSdpMediaDescr* field value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpMediaDescrGetBadSyntaxValue(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the bad syntax value.

rvSdpMediaDescrSetBadSyntax()

DESCRIPTION

Sets the SDP RvSdpMediaDescr field value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetBadSyntax(
    RvSdpMediaDescr*    o,
    const char*         bs);
```

PARAMETERS

o

A pointer to the *RvSdpMediaDescr.*

bs

The proprietary formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfFormats()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* codec formats.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfFormats(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined codec formats.

rvSdpMediaDescrGetFormat()

DESCRIPTION

Gets an *RvSdpMediaDescr* format by index.

SYNTAX

```
const char* rvSdpMediaDescrGetFormat(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfFormats().

RETURN VALUES

Returns the name of the requested codec format.

rvSdpMediaDescrGetNumOfPayloads()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* payloads.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfPayloads(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined payloads.

rvSdpMediaDescrGetPayload()

DESCRIPTION

Gets an *RvSdpMediaDescr* payload by index.

SYNTAX

```
int rvSdpMediaDescrGetPayload(
    RvSdpMediaDescr*    descr,
    int                 index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfPayloads().

RETURN VALUES

Returns the requested *RvSdpMediaDescr* payload.

rvSdpMediaDescrGetMediaType()

DESCRIPTION

Gets the media type of the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpMediaType rvSdpMediaDescrGetMediaType(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the media type value.

rvSdpMediaDescrSetMediaType()

DESCRIPTION

Sets the media type of the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrSetMediaType(
    RvSdpMediaDescr*    descr,
    RvSdpMediaType      type);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

type

The new media type.

RETURN VALUES

None.

rvSdpMediaDescrGetMediaTypeStr()

DESCRIPTION

Gets the media type string of the *RvSdpMediaDescr.*

SYNTAX

```
const char* rvSdpMediaDescrGetMediaTypeStr(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the media type text value.

rvSdpMediaDescrSetMediaTypeStr()

DESCRIPTION

Sets the media type string of the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetMediaTypeStr(
    RvSdpMediaDescr*    descr,
    const char*         type);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

type

The new value of the media type string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetProtocol()

DESCRIPTION

Gets the protocol of the *RvSdpMediaDescr.*

SYNTAX

```
RvSdpProtocol rvSdpMediaDescrGetProtocol(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the *RvSdpMediaDescr* protocol value.

rvSdpMediaDescrSetProtocol()

DESCRIPTION

Sets the protocol type of the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrSetProtocol(
    RvSdpMediaDescr*    descr,
    RvSdpProtocol       protocol);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

protocol

The new media protocol value.

RETURN VALUES

None.

rvSdpMediaDescrGetProtocolStr()

DESCRIPTION

Gets the media protocol name string of the *RvSdpMediaDescr*.

SYNTAX

```
const char* rvSdpMediaDescrGetProtocolStr(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the media protocol text value.

rvSdpMediaDescrSetProtocolStr()

DESCRIPTION

Sets the media protocol name string of the *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetProtocolStr(
    RvSdpMediaDescr*    descr,
    const char*         protocol);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

protocol

The new value of the media type string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetPort()

DESCRIPTION

Gets the media port number.

SYNTAX

```
RvUint32 rvSdpMediaDescrGetPort(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the port number.

rvSdpMediaDescrSetPort()

DESCRIPTION

Sets the media port number.

SYNTAX

```
void rvSdpMediaDescrSetPort(
    RvSdpMediaDescr*    descr,
    RvUint32            port);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

port

The new value of the media port number.

RETURN VALUES

None.

rvSdpMediaDescrGetNumOfPorts()

DESCRIPTION

Gets the number of subsequent ports defined for the *RvSdpMediaDescr.*

SYNTAX

```
int rvSdpMediaDescrGetNumOfPorts(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined subsequent ports.

rvSdpMediaDescrSetNumOfPorts()

DESCRIPTION

Sets the number of subsequent ports defined for the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrSetNumOfPorts(
    RvSdpMediaDescr*    descr,
    int                 numPorts);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

numPorts

The new number of subsequent ports.

RETURN VALUES

None.

rvSdpMediaDescrGetInformation()

DESCRIPTION

Gets the information field of the *RvSdpMediaDescr*.

SYNTAX

```
const char* rvSdpMediaDescrGetInformation(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the *RvSdpMediaDescr* information field text of the empty string if the information field is not set.

rvSdpMediaDescrSetInformation()

DESCRIPTION

Sets the information field of the *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetInformation(
    RvSdpMediaDescr*    descr,
    const char*         info);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

info

The new information value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfConnections()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* connection fields.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfConnections(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined connections.

rvSdpMediaDescrGetFirstConnection()

DESCRIPTION

Returns the first *RvSdpConnection* defined in the *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpConnection* rvSdpMediaDescrGetFirstConnection(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextConnection() calls.

RETURN VALUES

A pointer to the *RvSdpConnection*, or a NULL pointer if there are no connections defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextConnection()

DESCRIPTION

Returns the next *RvSdpConnection* defined in the *RvSdpMediaDescr*. The next *RvSdpConnection* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpConnection* rvSdpMediaDescrGetNextConnection(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstConnection() or rvSdpMediaDescrGetNextConnection() function.

RETURN VALUES

A pointer to the *RvSdpConnection*, or a NULL pointer if there are no more *RvSdpConnection* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetConnection()

DESCRIPTION

Gets the first *RvSdpConnection* (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpConnection* rvSdpMediaDescrGetConnection(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

The first *RvSdpConnection*, or NULL if there are no *RvSdpConnection* objects.

rvSdpMediaDescrGetConnectionByIndex()

DESCRIPTION

Gets an *RvSdpConnection* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpConnection* rvSdpMediaDescrGetConnectionByIndex(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfConnections().

RETURN VALUES

Returns the requested *RvSdpConnection.*

rvSdpMediaDescrSetConnection()

DESCRIPTION

Adds a new *RvSdpConnection* to an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetConnection(
    RvSdpMediaDescr*    descr,
    RvSdpNetType        net_type,
    RvSdpAddrType       addr_type,
    const char*         addr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

net_type

The network type.

addr_type

The address type.

addr

The address, depending on the network type. For example, an IP address for an IP network, and so on.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrSetBadSyntaxConnection()

DESCRIPTION

Adds an SDP *RvSdpConnection* field with a proprietary format for a specific *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetBadSyntaxConnection(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr*.

badSyn

The proprietary formatted *RvSdpConnection* field to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfBandwidth()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* bandwidth fields.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfBandwidth(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined bandwidths.

rvSdpMediaDescrGetFirstBandwidth()

DESCRIPTION

Returns the first bandwidth defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpBandwidth* rvSdpMediaDescrGetFirstBandwidth(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextBandwidth() calls.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth*, or a NULL pointer if there are no *RvSdpBandwidth* objects defined in the *RvSdpMediaDescr.*

rvSdpMediaDescrGetNextBandwidth()

DESCRIPTION

Returns the next *RvSdpBandwidth* defined in an *RvSdpMediaDescr*. The next *RvSdpBandwidth* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpBandwidth* rvSdpMediaDescrGetNextBandwidth(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstBandwidth() or rvSdpMediaDescrGetNextBandwidth() function.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth*, or a NULL pointer if there are no more *RvSdpBandwidth* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetBandwidth()

DESCRIPTION

Gets a first *RvSdpBandwidth* at the media-descriptor level.

SYNTAX

```
RvSdpBandwidth* rvSdpMediaDescrGetBandwidth(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the first *RvSdpBandwidth*, or NULL if there are no *RvSdpBandwidth* objects.

rvSdpMediaDescrGetBandwidthByIndex()

DESCRIPTION

Gets an *RvSdpBandwidth* by index at the media-descriptor level.

SYNTAX

```
RvSdpBandwidth* rvSdpMediaDescrGetBandwidthByIndex(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfBandwidth().

RETURN VALUES

Returns the requested *RvSdpBandwidth*.

rvSdpMediaDescrSetBandwidth()

DESCRIPTION

Adds a new *RvSdpBandwidth* at the media-descriptor level.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetBandwidth(
    RvSdpMediaDescr*    descr,
    const char*         bwtype,
    int                 b);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

bwtype

The bandwidth type, such as Conference Total (CT) or Application-Specific Maximum (AS).

b

The *RvSdpBandwidth* value in kilobits per second (kbps).

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrSetBadSyntaxBandwidth()

DESCRIPTION

Adds the SDP bandwidth field with a proprietary format for a specific *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetBadSyntaxBandwidth(
    RvSdpMediaDescr*    descr,
    const char*         bandwidth);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr*.

bandwidth

The proprietary formatted *RvSdpBandwidth* field to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetKey()

DESCRIPTION

Gets a pointer to the key field of an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpKey* rvSdpMediaDescrGetKey(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to valid *RvSdpMediaDescr.*

RETURN VALUES

Returns a pointer to the key field, or NULL if the key field is not set in the RvSdpMediaDescr.

rvSdpMediaDescrSetKey()

DESCRIPTION

Sets the SDP key field in an *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetKey(
    RvSdpMediaDescr*    descr,
    RvSdpEncrMethod     em,
    const char*         key);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr*.

em

The key encryption method.

key

The key value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrSetBadSyntaxKey()

DESCRIPTION

Sets the SDP key field with a proprietary format for a specific *RvSdpMediaDescr*.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetBadSyntaxKey(
    RvSdpMediaDescr*    descr,
    const char*         badSyn);
```

PARAMETERS

descr

A pointer to a valid *RvSdpMediaDescr*.

badSyn

The proprietary formatted key field to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfAttr()

DESCRIPTION

Gets the number of *RvSdpMediaDescr* generic attributes.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfAttr(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined generic attributes.

rvSdpMediaDescrGetFirstAttribute()

DESCRIPTION

Returns the first generic *RvSdpAttribute* defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetFirstAttribute(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextAttribute() calls.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no generic *RvSdpAttribute* defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextAttribute()

DESCRIPTION

Returns the next generic *RvSdpAttribute* defined in an *RvSdpMediaDescr.* The next *RvSdpAttribute* is defined based on the state of the *RvSdpListIter.*

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetNextAttribute(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to thervSdpMediaDescrGetFirstAttribute() or rvSdpMediaDescrGetNextAttribute() function.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no more generic *RvSdpAttribute* objects defined in the *RvSdpMediaDescr.*

rvSdpMediaDescrGetAttribute()

DESCRIPTION

Gets a generic *RvSdpAttribute* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetAttribute(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfAttr().

RETURN VALUES

Returns the requested *RvSdpAttribute* pointer.

rvSdpMediaDescrGetNumOfAttr2()

DESCRIPTION

Gets the number of *RvSdpAttribute* objects (generic and special) of the *RvSdpMediaDescr*.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfAttr2(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the number of defined *RvSdpAttribute* objects.

rvSdpMediaDescrGetFirstAttribute2()

DESCRIPTION

Returns the first *RvSdpAttribute* (generic or special) defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetFirstAttribute2(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextAttribute2() calls.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no *RvSdpAttribute* objects defined in the *RvSdpMediaDescr.*

rvSdpMediaDescrGetNextAttribute2()

DESCRIPTION

Returns the next *RvSdpAttribute* (generic or special) defined in an *RvSdpMediaDescr*. The next *RvSdpAttribute* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetNextAttribute2(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstAttribute2() or rvSdpMediaDescrGetNextAttribute2() function.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute*, or a NULL pointer if there are no more generic *RvSdpAttribute* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetAttribute2()

DESCRIPTION

Gets an *RvSdpAttribute* (generic or special) by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetAttribute2(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfAttr2() call.

RETURN VALUES

Returns the requested *RvSdpAttribute* pointer.

rvSdpMediaDescrGetNumOfRtpMap()

DESCRIPTION

Gets the number of *RvSdpRtpMap* attributes of an *RvSdpMediaDescr.*

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfRtpMap(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined codec *RvSdpRtpMap* objects.

rvSdpMediaDescrGetFirstRtpMap()

DESCRIPTION

Returns the first *RvSdpRtpMap* defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpRtpMap* rvSdpMediaDescrGetFirstRtpMap(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextRtpMap() calls.

RETURN VALUES

Returns a pointer to the *RvSdpRtpMap*, or a NULL pointer if there are no *RvSdpRtpMap* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextRtpMap()

DESCRIPTION

Returns the next *RvSdpRtpMap* defined in an *RvSdpMediaDescr*. The next *RvSdpRtpMap* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpRtpMap* rvSdpMediaDescrGetNextRtpMap(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to rrvSdpMediaDescrGetFirstRtpMap() function.

RETURN VALUES

Returns a pointer to the *RvSdpRtpMap*, or a NULL pointer if there are no more *RvSdpRtpMap* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetRtpMap()

DESCRIPTION

Gets an *RvSdpRtpMap* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpRtpMap* rvSdpMediaDescrGetRtpMap(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfRtpMap().

RETURN VALUES

Returns the requested *RvSdpRtpMap* pointer.

rvSdpMediaDescrGetConnectionMode()

DESCRIPTION

Gets the connection mode of an *RvSdpMediaDescr* or RV_SDPCONNECTMODE_NOTSET if the correspondent attribute is not set.

SYNTAX

```
RvSdpConnectionMode rvSdpMediaDescrGetConnectionMode(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the connection mode.

rvSdpMediaDescrSetConnectionMode()

DESCRIPTION

Sets or modifies the connection mode of an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetConnectionMode(
    RvSdpMediaDescr*       descr,
    RvSdpConnectionMode    mode);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

mode

The new value of connection mode.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfKeyMgmt()

DESCRIPTION

Gets the number of *RvSdpKeyMgmtAttr* objects of an *RvSdpMediaDescr.*

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfKeyMgmt(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined codec *RvSdpKeyMgmtAttr* objects.

rvSdpMediaDescrGetFirstKeyMgmt()

DESCRIPTION

Returns the first key *RvSdpKeyMgmtAttr* defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMediaDescrGetFirstKeyMgmt(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextKeyMgmt() calls.

RETURN VALUES

Returns a pointer to the *RvSdpKeyMgmtAttr*, or a NULL pointer if there are no *RvSdpKeyMgmtAttr* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextKeyMgmt()

DESCRIPTION

Returns the next *RvSdpKeyMgmtAttr* defined in an *RvSdpMediaDescr*. The next *RvSdpKeyMgmtAttr* is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMediaDescrGetNextKeyMgmt(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstKeyMgmt() or rvSdpMediaDescrGetNextKeyMgmt() function.

RETURN VALUES

Returns a pointer to the *RvSdpKeyMgmtAttr*, or a NULL pointer if there are no more *RvSdpKeyMgmtAttr* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetKeyMgmt()

DESCRIPTION

Gets an *RvSdpKeyMgmtAttr* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpKeyMgmtAttr* rvSdpMediaDescrGetKeyMgmt(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfKeyMgmt().

RETURN VALUES

Returns the requested *RvSdpKeyMgmtAttr* pointer.

rvSdpMediaDescrGetNumOfCrypto()

DESCRIPTION

Gets the number of *RvSdpCryptoAttr* objects set in the context of an *RvSdpMediaDescr*.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfCrypto(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the number of the defined *RvSdpCryptoAttr*.

rvSdpMediaDescrGetFirstCrypto()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* to which the *iter* parameter points in the context of an *RvSdpMediaDescr*. The value of *iter* is undefined after the function call.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMediaDescrGetFirstCrypto(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextCrypto() calls.

RETURN VALUES

Returns a pointer to the *RvSdpCryptoAttr*, or a NULL pointer if there are no crypto attributes defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextCrypto()

DESCRIPTION

Returns the next *RvSdpCryptoAttr* defined in an *RvSdpMediaDescr*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpCryptoAttr* rvSdpMediaDescrGetNextCrypto(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstCrypto() function.

RETURN VALUES

Returns a pointer to the *RvSdpCryptoAttr*, or a NULL pointer if there are no more Crypto attributes defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetCrypto()

DESCRIPTION

Gets an *RvSdpCryptoAttr* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpCryptoAttr* rvSdpMediaDescrGetCrypto(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfCrypto() call.

RETURN VALUES

Returns the requested *RvSdpCryptoAttr* pointer.

rvSdpMediaDescrGetFrameRate()

DESCRIPTION

Gets the frame-rate special attribute value of an *RvSdpMediaDescr*.

SYNTAX

```
const char* rvSdpMediaDescrGetFrameRate(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the value of the attribute.

rvSdpMediaDescrSetFrameRate()

DESCRIPTION

Sets or modifies the frame-rate special attribute of an *RvSdpMediaDescr.*

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetFrameRate(
    RvSdpMediaDescr*    descr,
    const char*         val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The framerate attribute value.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfFmtp()

DESCRIPTION

Gets the number *fmtp* special attributes of an *RvSdpMediaDescr*.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfFmtp(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the number of defined *fmtp* special attributes.

rvSdpMediaDescrGetFirstFmtp()

DESCRIPTION

Returns the first *fmtp* special attribute defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetFirstFmtp(
    RvSdpMediaDescr*    descr,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextFmtp() calls.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute* (of *fmtp*) object, or a NULL pointer if there are no *fmtp* attributes defined in the *RvSdpMediaDescr.*

rvSdpMediaDescrGetNextFmtp()

DESCRIPTION

Returns the next *fmtp* special attribute defined in an *RvSdpMediaDescr*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetNextFmtp(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstFmtp() or rvSdpMediaDescrGetNextFmtp() function.

RETURN VALUES

Returns a pointer to the *RvSdpAttribute* (*fmtp*), or a NULL pointer if there is no more *fmtp* attributes defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetFmtp()

DESCRIPTION

Gets an *fmtp* special attribute by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpAttribute* rvSdpMediaDescrGetFmtp(
    const RvSdpMediaDescr*    descr,
    RvSize_t                  index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfFmtp().

RETURN VALUES

Returns the requested *RvSdpAttribute* (of *fmtp* special attribute) pointer.

rvSdpMediaDescrGetNumOfOther()

DESCRIPTION

Gets the number of *RvSdpOther* objects in an *RvSdpMediaDescr.*

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfOther(
    const RvSdpMediaDescr*    media);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the number of defined *RvSdpOther* objects.

rvSdpMediaDescrGetFirstOther()

DESCRIPTION

Returns the first *RvSdpOther* defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpOther* rvSdpMediaDescrGetFirstOther(
    RvSdpMediaDescr*    media,
    RvSdpListIter*      iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextOther() calls.

RETURN VALUES

Returns a pointer to the *RvSdpOther*, or a NULL pointer if there are no *RvSdpOther* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextOther()

DESCRIPTION

Returns the next *RvSdpOther* defined in an *RvSdpMediaDescr*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpOther* rvSdpMediaDescrGetNextOther(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstOther() or rvSdpMediaDescrGetNextOther() function.

RETURN VALUES

Returns a pointer to the *RvSdpOther*, or a NULL pointer if there are no more *RvSdpOther* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetOther()

DESCRIPTION

Gets an *RvSdpOther* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpOther* rvSdpMediaDescrGetOther(
    RvSdpMediaDescr*    media,
    RvSize_t            index);
```

PARAMETERS

media

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfOther().

RETURN VALUES

Returns the requested *RvSdpOther.*

rvSdpMediaDescrGetNumOfBadSyntax()

DESCRIPTION

Gets the number of *RvSdpBadSyntax* objects of an *RvSdpMediaDescr*.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfBadSyntax(
    const RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

Returns the number of defined *RvSdpBadSyntax* objects.

rvSdpMediaDescrGetFirstBadSyntax()

DESCRIPTION

Returns the first *RvSdpBadSyntax* defined in an *RvSdpMediaDescr*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpBadSyntax* rvSdpMediaDescrGetFirstBadSyntax(
    RvSdpMediaDescr*     descr,
    RvSdpLineObjIter*    iter);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMediaDescrGetNextBadSyntax() calls.

RETURN VALUES

Returns a pointer to the *RvSdpBadSyntax*, or a NULL pointer if there are no *RvSdpBadSyntax* objects defined in the *RvSdpMediaDescr.*

rvSdpMediaDescrGetNextBadSyntax()

DESCRIPTION

Returns the next *RvSdpBadSyntax* defined in an *RvSdpMediaDescr*. The next object is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpBadSyntax* rvSdpMediaDescrGetNextBadSyntax(
    RvSdpLineObjIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstBadSyntax() or rvSdpMediaDescrGetNextBadSyntax() function.

RETURN VALUES

Returns a pointer to the *RvSdpBadSyntax*, or a NULL pointer if there are no more *RvSdpBadSyntax* objects defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetBadSyntax()

DESCRIPTION

Gets an *RvSdpBadSyntax* by index (in the context of an *RvSdpMediaDescr*).

SYNTAX

```
RvSdpBadSyntax* rvSdpMediaDescrGetBadSyntax(
    const RvSdpMediaDescr*    descr,
    int                       index);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpMediaDescrGetNumOfBadSyntax() call.

RETURN VALUES

Returns the requested *RvSdpBadSyntax*.

6

MESSAGE LIST FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains functions for operating on *RvSdpMsgList* objects.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMsgListConstruct()
- rvSdpMsgListConstructA()
- rvSdpMsgListConstructCopyA()
- rvSdpMsgListDestruct()
- rvSdpMsgListCopy()
- rvSdpMsgListAddMsg()
- rvSdpMsgListInsertMsg()
- rvSdpMsgListAppendMsg()
- rvSdpMsgListRemoveCurrentMsg()
- rvSdpMsgListRemoveElement()
- rvSdpMsgListClear()

rvSdpMsgListConstruct()

DESCRIPTION

Constructs a list of RvSdpMsgs.

SYNTAX

```
RvSdpMsgList* rvSdpMsgListConstruct(
    RvSdpMsgList*    msgList);
```

PARAMETERS

msgList

A pointer to a valid RvSdpMsgList.

RETURN VALUES

Returns the constructed RvSdpMsgList, or NULL if the function fails.

rvSdpMsgListConstructA()

DESCRIPTION

Constructs a list of RvSdpMsgs using an *RvAlloc* that the user provides.

SYNTAX

```
RvSdpMsgList* rvSdpMsgListConstructA(
    RvSdpMsgList*    msgList,
    RvAlloc*         a);
```

PARAMETERS

msgList

A pointer to a valid RvSdpMsgList.

a

The *RvAlloc* to be used.

RETURN VALUES

Returns the constructed RvSdpMsgList, or NULL if the function fails.

rvSdpMsgListConstructCopyA()

DESCRIPTION

Constructs an RvSdpMsgList and copies all RvSdpMsgs contained in the source RvSdpMsgList to the destination RvSdpMsgList.

SYNTAX

```
RvSdpMsgList* rvSdpMsgListConstructCopyA(
    RvSdpMsgList*          dest,
    const RvSdpMsgList*    src,
    RvAlloc*               a);
```

PARAMETERS

dest

The destination RvSdpMsgList.

src

The source RvSdpMsgList

a

The *RvAlloc* to be used.

RETURN VALUES

Returns the constructed destination, or NULL if the function fails.

rvSdpMsgListDestruct()

DESCRIPTION

Destructs all RvSdpMsgs contained in *msgList*.

SYNTAX

```
void rvSdpMsgListDestruct(
    RvSdpMsgList*    msgList);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

RETURN VALUES

None.

rvSdpMsgListCopy()

DESCRIPTION

Copies all RvSdpMsgs contained in the *src* to the *dest*.

SYNTAX

```
RvSdpMsgList* rvSdpMsgListCopy(
    RvSdpMsgList*          dest,
    const RvSdpMsgList*    src);
```

PARAMETERS

dest

The destination of the RvSdpMsgList.

src

The source of the RvSdpMsgList.

RETURN VALUES

Returns the destination RvSdpMsgList, or NULL if the function fails.

rvSdpMsgListAddMsg()

DESCRIPTION

Constructs a new RvSdpMsg and adds it to an RvSdpMsgList.

SYNTAX

```
RvSdpMsg *rvSdpMsgListAddMsg(
    RvSdpMsgList*    msgList);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

RETURN VALUES

Returns a pointer to the constructed or added RvSdpMsg, or NULL if the function fails.

rvSdpMsgListInsertMsg()

DESCRIPTION

Adds a new element to the list by copying the values of an existing RvSdpMsg.

SYNTAX

```
RvSdpStatus rvSdpMsgListInsertMsg(
    RvSdpMsgList*      msgList,
    const RvSdpMsg*    msg);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

msg

A pointer to the constructed RvSdpMsg.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgListAppendMsg()

DESCRIPTION

Appends the constructed and valid SDP message to the list.

SYNTAX

```
void rvSdpMsgListAppendMsg(
    RvSdpMsgList*      msgList,
    const RvSdpMsg*    msg)
```

PARAMETERS

msgList

The RvSdpMsgList object to be destructed.

msg

A pointer to the constructed RvSdpMsg object.

RETURN VALUES

None.

rvSdpMsgListRemoveCurrentMsg()

DESCRIPTION

Removes (and destructs) an RvSdpMsg to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpMsgListRemoveCurrentMsg(
    RvSdpListIter*    li);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgListGetFirstMsg() or rvSdpMsgListGetNextMsg() function.

RETURN VALUES

None.

rvSdpMsgListRemoveElement()

DESCRIPTION

Removes and destructs the RvSdpMsg contained in the list by index.

SYNTAX

```
void rvSdpMsgListRemoveElement(
    RvSdpMsgList*    msgList,
    RvSize_t         i);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

i

The index of the RvSdpMsg to be removed.

RETURN VALUES

None.

rvSdpMsgListClear()

DESCRIPTION

Removes (and destructs) all RvSdpMsg objects set in the message list.

SYNTAX

```
void rvSdpMsgListClear(
    RvSdpMsgList*    msgList);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpMsgListGetSize()
- rvSdpMsgListGetFirstMsg()
- rvSdpMsgListGetNextMsg()
- rvSdpMsgListGetElement()

rvSdpMsgListGetSize()

DESCRIPTION

Gets the number of RvSdpMsg objects in an RvSdpMsgList.

SYNTAX

```
RvSize_t rvSdpMsgListGetSize(
    const RvSdpMsgList*    msgList);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

RETURN VALUES

Returns the number of RvSdpMsg objects in the RvSdpMsgList.

rvSdpMsgListGetFirstMsg()

DESCRIPTION

Returns the first RvSdpMsg in an RvSdpMsgList. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpMsg * rvSdpMsgListGetFirstMsg(
    RvSdpMsgList*    msgList,
    RvSdpListIter*   li);
```

PARAMETERS

msgList

A pointer to the RvSdpMsgList.

li

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpMsgListGetNextMsg() calls.

RETURN VALUES

Returns a pointer to the RvSdpMsgList, or a NULL pointer if there are no messages in the list.

rvSdpMsgListGetNextMsg()

DESCRIPTION

Returns the next RvSdpMsg from an RvSdpMsgList. The next RvSdpMsg is defined based on the state of the *RvSdpListIter*.

SYNTAX

```
RvSdpMsg* rvSdpMsgListGetNextMsg(
    RvSdpListIter*    li);
```

PARAMETERS

li

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgListGetFirstMsg() or rvSdpMsgListGetNextMsg() function.

RETURN VALUES

Returns a pointer to the RvSdpMsg, or a NULL pointer if there are no more RvSdpMsg objects in the list.

rvSdpMsgListGetElement()

DESCRIPTION

Gets an RvSdpMsg with the “i” index contained in the *msgList*.

SYNTAX

```
RvSdpMsg* rvSdpMsgListGetElement(
    RvSdpMsgList*    msgList,
    RvSize_t         i);
```

PARAMETERS

msgList

The RvSdpMsgList to be destructed.

RETURN VALUES

Returns the RvSdpMsg with the “i” index contained in the *msgList*.

7

ORIGIN FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions for operating on *RvSdpOrigin* objects. The RvSdpOrigin type represents the origin ('o=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpOriginConstruct() **Obsolete**
- rvSdpOriginConstructA() **Obsolete**
- rvSdpOriginConstructCopy() **Obsolete**
- rvSdpOriginConstructCopyA() **Obsolete**
- rvSdpBadSyntaxOriginConstruct() **Obsolete**
- rvSdpBadSyntaxOriginConstructA() **Obsolete**
- rvSdpOriginIsBadSyntax()
- rvSdpOriginDestruct()
- rvSdpOriginCopy()

rvSdpOriginConstruct()

DESCRIPTION

Constructs an *RvSdpOrigin*.

This function is obsolete. Please use rvSdpMsgSetOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpOriginConstruct(
    RvSdpOrigin      *origin,
    const char*      username,
    const char*      session_id,
    const char*      version,
    RvSdpNetType     nettype,
    RvSdpAddrType    addrtype,
    const char*      address);
```

PARAMETERS

origin

A pointer to a valid *RvSdpOrigin*.

username

The field user name.

session_id

The field session ID.

version

The field version.

nettype

The network type.

addrtype

The address type.

addr

The connection address.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOrigin*, or NULL if the function fails.

rvSdpOriginConstructA()

DESCRIPTION

Constructs an *RvSdpOrigin*.

This function is obsolete. Please use rvSdpMsgSetOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpOriginConstructA(
    RvSdpOrigin     *origin,
    const char*      username,
    const char*      session_id,
    const char*      version,
    RvSdpNetType     nettype,
    RvSdpAddrType    addrtype,
    const char*      address,
    RvAlloc*         a);
```

PARAMETERS

origin

A pointer to the valid.

username

The field user name.

session_id

The field session ID.

version

The field version.

nettype

The network type.

addrtype

The address type.

addr

The connection address.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOrigin*, or NULL if the function fails.

rvSdpOriginConstructCopy()

DESCRIPTION

Constructs an *RvSdpOrigin* and copies the values from a source *RvSdpOrigin* field.

This function is obsolete. Please use rvSdpMsgSetOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpOriginConstructCopy(
    RvSdpOrigin*          dest,
    const RvSdpOrigin*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpOrigin* to be constructed. This parameter must point to valid memory.

src

A source *RvSdpOrigin*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOrigin*, or NULL if the function fails.

rvSdpOriginConstructCopyA()

DESCRIPTION

Constructs an *RvSdpOrigin* and copies the values from a source *RvSdpOrigin* field.

This function is obsolete. Please use rvSdpMsgSetOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpOriginConstructCopyA(
    RvSdpOrigin*          dest,
    const RvSdpOrigin*    src,
    RvAlloc*              a);
```

PARAMETERS

dest

A pointer to an *RvSdpOrigin* to be constructed. This parameter must point to valid memory.

src

The *RvSdpOrigin*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed object, or NULL if the function fails.

rvSdpBadSyntaxOriginConstruct()

DESCRIPTION

Constructs an *RvSdpOrigin* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpBadSyntaxOriginConstruct(
    RvSdpOrigin*    origin,
    const char*     badSyn);
```

PARAMETERS

origin

A pointer to a valid *RvSdpOrigin*.

badSyn

The proprietary format of the *RvSdpOrigin* field.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOrigin*, or NULL if the function fails.

rvSdpBadSyntaxOriginConstructA()

DESCRIPTION

Constructs an *RvSdpOrigin* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxOrigin() instead.

SYNTAX

```
RvSdpOrigin* rvSdpBadSyntaxOriginConstructA(
    RvSdpOrigin*    origin,
    const char*     badSyn,
    RvAlloc*        a);
```

PARAMETERS

origin

A pointer to valid *RvSdpOrigin*.

badSyn

The proprietary format of the *RvSdpOrigin* field.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOrigin*, or NULL if the function fails.

rvSdpOriginIsBadSyntax()

DESCRIPTION

Tests whether the *RvSdpOrigin* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpOriginIsBadSyntax(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpOriginDestruct()

DESCRIPTION

Destructs an *RvSdpOrigin*.

SYNTAX

```
void rvSdpOriginDestruct(
    RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

None.

rvSdpOriginCopy()

DESCRIPTION

Copies the values from a source *RvSdpOrigin* to a destination *RvSdpOrigin*.

SYNTAX

```
RvSdpOrigin* rvSdpOriginCopy(
    RvSdpOrigin*          dest,
    const RvSdpOrigin*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpOrigin*. This parameter must point to a constructed *RvSdpOrigin*.

src

A source *RvSdpOrigin*.

RETURN VALUES

Returns a pointer to the destination *RvSdpOrigin*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpOriginGetBadSyntax()
- rvSdpOriginSetBadSyntax()
- rvSdpOriginGetUsername()
- rvSdpOriginSetUsername()
- rvSdpOriginGetVersion()
- rvSdpOriginSetVersion()
- rvSdpOriginGetSessionId()
- rvSdpOriginSetSessionId()
- rvSdpOriginGetNetType()
- rvSdpOriginSetNetType()
- rvSdpOriginGetNetTypeStr()
- rvSdpOriginSetNetTypeStr()
- rvSdpOriginGetAddressType()
- rvSdpOriginSetAddressType()
- rvSdpOriginGetAddressTypeStr()
- rvSdpOriginSetAddressTypeStr()
- rvSdpOriginGetAddress()
- rvSdpOriginSetAddress()

rvSdpOriginGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpOrigin* field value, or empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpOriginGetBadSyntax(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the bad syntax value.

rvSdpOriginSetBadSyntax()

DESCRIPTION

Sets the *RvSdpOrigin* field value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpOriginSetBadSyntax(
    RvSdpOrigin*    o,
    const char*     bs);
```

PARAMETERS

o

A pointer to the *RvSdpOrigin*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetUsername()

DESCRIPTION

Gets the user name of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetUsername(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the user name of the *RvSdpOrigin*.

rvSdpOriginSetUsername()

DESCRIPTION

Sets the user name of the *RvSdpOrigin* field.

SYNTAX

```
RvSdpStatus rvSdpOriginSetUsername(
    RvSdpOrigin*    origin,
    const char*     username);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

username

The user name of the *RvSdpOrigin* field.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetVersion()

DESCRIPTION

Gets the version of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetVersion(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the version of the *RvSdpOrigin*.

rvSdpOriginSetVersion()

DESCRIPTION

Sets the version of an *RvSdpOrigin* field.

SYNTAX

```
RvSdpStatus rvSdpOriginSetVersion(
    RvSdpOrigin*    origin,
    const char*     version);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

version

The new version value of an *RvSdpOrigin* field.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetSessionId()

DESCRIPTION

Gets the session ID of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetSessionId(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the session ID of the *RvSdpOrigin*.

rvSdpOriginSetSessionId()

DESCRIPTION

Sets the field session ID of an *RvSdpOrigin*.

SYNTAX

```
RvSdpStatus rvSdpOriginSetSessionId(
    RvSdpOrigin*    origin,
    const char*     id);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

id

The field session ID of the *RvSdpOrigin*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetNetType()

DESCRIPTION

Gets the network type of an *RvSdpOrigin*.

SYNTAX

```
RvSdpNetType rvSdpOriginGetNetType(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the network type of the *RvSdpOrigin*.

rvSdpOriginSetNetType()

DESCRIPTION

Sets the network type field of an *RvSdpOrigin*.

SYNTAX

```
void rvSdpOriginSetNetType(
    RvSdpOrigin*    origin,
    RvSdpNetType    netType);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

netType

The new value of the network type field of the *RvSdpOrigin*.

RETURN VALUES

None.

rvSdpOriginGetNetTypeStr()

DESCRIPTION

Gets the network type string of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetNetTypeStr(
    RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the network type string of the *RvSdpOrigin*.

rvSdpOriginSetNetTypeStr()

DESCRIPTION

Sets the field network type string of an *RvSdpOrigin*.

SYNTAX

```
RvSdpStatus rvSdpOriginSetNetTypeStr(
    RvSdpOrigin*    origin,
    const char*     type);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

type

The field network type string of the *RvSdpOrigin*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetAddressType()

DESCRIPTION

Gets the address type of an *RvSdpOrigin*.

SYNTAX

```
RvSdpAddrType rvSdpOriginGetAddressType(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the address type of the *RvSdpOrigin*.

rvSdpOriginSetAddressType()

DESCRIPTION

Sets the field address type of an *RvSdpOrigin*.

SYNTAX

```
void rvSdpOriginSetAddressType(
    RvSdpOrigin*     origin,
    RvSdpAddrType    addrType);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

addrType

The field address type of the *RvSdpOrigin*.

RETURN VALUES

None.

rvSdpOriginGetAddressTypeStr()

DESCRIPTION

Gets the address type string of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetAddressTypeStr(
    RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the address type string of the *RvSdpOrigin*.

rvSdpOriginSetAddressTypeStr()

DESCRIPTION

Sets the field address type string of an *RvSdpOrigin*.

SYNTAX

```
RvSdpStatus rvSdpOriginSetAddressTypeStr(
    RvSdpOrigin*    origin,
    const char*     t);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

t

The field address type string of the *RvSdpOrigin*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpOriginGetAddress()

DESCRIPTION

Gets the address of an *RvSdpOrigin*.

SYNTAX

```
const char* rvSdpOriginGetAddress(
    const RvSdpOrigin*    origin);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

RETURN VALUES

Returns the address of the *RvSdpOrigin*.

rvSdpOriginSetAddress()

DESCRIPTION

Sets the field address of an *RvSdpOrigin*.

SYNTAX

```
RvSdpStatus rvSdpOriginSetAddress(
    RvSdpOrigin*    origin,
    const char*     addr);
```

PARAMETERS

origin

A pointer to the *RvSdpOrigin*.

addr

The field address of the *RvSdpOrigin*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

8

EMAIL FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions for operating on *RvSdpEmail* objects. The RvSdpEmail type represents the email ('e=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpEmailConstruct() **Obsolete**
- rvSdpEmailConstructA() **Obsolete**
- rvSdpEmailConstructCopy() **Obsolete**
- rvSdpEmailConstructCopyA() **Obsolete**
- rvSdpBadSyntaxEmailConstruct() **Obsolete**
- rvSdpBadSyntaxEmailConstructA()**Obsolete**
- rvSdpEmailIsBadSyntax()
- rvSdpEmailDestruct()
- rvSdpEmailCopy()

rvSdpEmailConstruct()

DESCRIPTION

Constructs an *RvSdpEmail*.

This function is obsolete. Please use rvSdpMsgAddEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpEmailConstruct(
    RvSdpEmail*    email,
    const char*    address,
    const char*    text);
```

PARAMETERS

email

A pointer to a valid *RvSdpEmail*.

address

The *RvSdpEmail* address.

text

The optional *RvSdpEmail* text.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpEmailConstructA()

DESCRIPTION

Constructs an *RvSdpEmail*.

This function is obsolete. Please use rvSdpMsgAddEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpEmailConstructA(
    RvSdpEmail*    email,
    const char*    address,
    const char*    text,
    RvAlloc*       a);
```

PARAMETERS

email

A pointer to a valid *RvSdpEmail*.

address

The *RvSdpEmail* address.

text

The optional *RvSdpEmail* text.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpEmailConstructCopy()

DESCRIPTION

Constructs an *RvSdpEmail* and copies the values from a source *RvSdpEmail* field.

This function is obsolete. Please use rvSdpMsgAddEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpEmailConstructCopy(
    RvSdpEmail*          dest,
    const RvSdpEmail*    source);
```

PARAMETERS

dest

A pointer to the *RvSdpEmail* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpEmail*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpEmailConstructCopyA()

DESCRIPTION

Constructs an *RvSdpEmail* and copies the values from a source *RvSdpEmail* field.

This function is obsolete. Please use rvSdpMsgAddEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpEmailConstructCopyA(
    RvSdpEmail*          dest,
    const RvSdpEmail*    source,
    RvAlloc*             a);
```

PARAMETERS

dest

A pointer to the *RvSdpEmail* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpEmail*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpBadSyntaxEmailConstruct()

DESCRIPTION

Constructs an *RvSdpEmail* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpBadSyntaxEmailConstruct(
    RvSdpEmail*    email,
    const char*    badSyn);
```

PARAMETERS

email

A pointer to a valid *RvSdpEmail*.

badSyn

The proprietary format of the *RvSdpEmail*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpBadSyntaxEmailConstructA()

DESCRIPTION

Constructs an *RvSdpEmail* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxEmail() instead.

SYNTAX

```
RvSdpEmail* rvSdpBadSyntaxEmailConstructA(
    RvSdpEmail*    email,
    const char*    badSyn,
    RvAlloc*       a);
```

PARAMETERS

email

A pointer to a valid *RvSdpEmail*.

badSyn

The proprietary format of the *RvSdpEmail*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpEmail*, or NULL if the function fails.

rvSdpEmailIsBadSyntax()

DESCRIPTION

Indicates whether or not the *RvSdpEmail* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpEmailIsBadSyntax(
    RvSdpEmail*    email);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpEmailDestruct()

DESCRIPTION

Destructs an *RvSdpEmail*.

SYNTAX

```
void rvSdpEmailDestruct(
    RvSdpEmail    *email);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

RETURN VALUES

None.

rvSdpEmailCopy()

DESCRIPTION

Copies the values from a source *RvSdpEmail* to a destination *RvSdpEmail*.

SYNTAX

```
RvSdpEmail* rvSdpEmailCopy(
    RvSdpEmail*          dest,
    const RvSdpEmail*    source);
```

PARAMETERS

dest

A pointer to the destination *RvSdpEmail*. This parameter must point to the constructed *RvSdpEmail*.

src

The source *RvSdpEmail*.

RETURN VALUES

Returns a pointer to the destination *RvSdpEmail*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpEmailGetBadSyntax()
- rvSdpEmailSetBadSyntax()
- rvSdpEmailGetAddress()
- rvSdpEmailSetAddress()
- rvSdpEmailGetText()
- rvSdpEmailSetText()

rvSdpEmailGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpEmail* field value, or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpEmailGetBadSyntax(
    const RvSdpEmail*    email);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

RETURN VALUES

Returns the bad syntax value.

rvSdpEmailSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpEmail* value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpEmailSetBadSyntax(
    RvSdpEmail*    o,
    const char*    bs);
```

PARAMETERS

o

A pointer to the *RvSdpEmail*.

bs

The proprietary formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpEmailGetAddress()

DESCRIPTION

Gets the *RvSdpEmail* address.

SYNTAX

```
const char* rvSdpEmailGetAddress(
    const RvSdpEmail*    email);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

RETURN VALUES

Returns the *RvSdpEmail* address.

rvSdpEmailSetAddress()

DESCRIPTION

Sets the *RvSdpEmail* address.

SYNTAX

```
RvSdpStatus rvSdpEmailSetAddress(
    RvSdpEmail*    email,
    const char*    addr);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

addr

The *RvSdpEmail* address.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpEmailGetText()

DESCRIPTION

Gets the optional text of the *RvSdpEmail*.

SYNTAX

```
const char* rvSdpEmailGetText(
    const RvSdpEmail*    email);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

RETURN VALUES

Returns the optional text of the *RvSdpEmail*.

rvSdpEmailSetText()

DESCRIPTION

Sets the *RvSdpEmail* text.

SYNTAX

```
RvSdpStatus rvSdpEmailSetText(
    RvSdpEmail*    email,
    const char*    text);
```

PARAMETERS

email

A pointer to the *RvSdpEmail*.

text

The *RvSdpEmail* text.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

9

PHONE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions for operating on *RvSdpPhone* objects. The RvSdpPhone type represents the phone ('p=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpPhoneConstruct() **Obsolete**
- rvSdpPhoneConstructA() **Obsolete**
- rvSdpPhoneConstructCopy() **Obsolete**
- rvSdpPhoneConstructCopyA() **Obsolete**
- rvSdpBadSyntaxPhoneConstruct() **Obsolete**
- rvSdpBadSyntaxPhoneConstructA() **Obsolete**
- rvSdpPhoneIsBadSyntax()
- rvSdpPhoneDestruct()
- rvSdpPhoneCopy()

rvSdpPhoneConstruct()

DESCRIPTION

Constructs an *RvSdpPhone*.

This function is obsolete. Please use rvSdpMsgAddPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpPhoneConstruct(
    RvSdpPhone*    phone,
    const char*    number,
    const char*    text);
```

PARAMETERS

phone

A pointer to a valid *RvSdpPhone*.

number

The *RvSdpPhone* number.

text

The optional *RvSdpPhone* text.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpPhoneConstructA()

DESCRIPTION

Constructs an *RvSdpPhone*.

This function is obsolete. Please use rvSdpMsgAddPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpPhoneConstructA(
    RvSdpPhone*    phone,
    const char*    number,
    const char*    text,
    RvAlloc*       alloc);
```

PARAMETERS

phone

A pointer to a valid *RvSdpPhone*.

number

The *RvSdpPhone* number.

text

The optional *RvSdpPhone* text.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpPhoneConstructCopy()

DESCRIPTION

Constructs an *RvSdpPhone* and copies the values from a source *RvSdpPhone* field.

This function is obsolete. Please use rvSdpMsgAddPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpPhoneConstructCopy(
    RvSdpPhone*          dest,
    const RvSdpPhone*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpPhone* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpPhone* object.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpPhoneConstructCopyA()

DESCRIPTION

Constructs an *RvSdpPhone* and copies the values from a source *RvSdpPhone* field.

This function is obsolete. Please use rvSdpMsgAddPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpPhoneConstructCopyA(
    RvSdpPhone*          dest,
    const RvSdpPhone*    src,
    RvAlloc*             alloc);
```

PARAMETERS

dest

A pointer to the *RvSdpPhone* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpPhone* object.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpBadSyntaxPhoneConstruct()

DESCRIPTION

Constructs an *RvSdpPhone* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpBadSyntaxPhoneConstruct(
    RvSdpPhone*    phone,
    const char*    badSyn);
```

PARAMETERS

phone

A pointer to a valid *RvSdpPhone*.

badSyn

The proprietary format of the *RvSdpPhone*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpBadSyntaxPhoneConstructA()

DESCRIPTION

Constructs an *RvSdpPhone* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxPhone() instead.

SYNTAX

```
RvSdpPhone* rvSdpBadSyntax
    PhoneConstructA(
    RvSdpPhone*    phone,
    const char*    badSyn,
    RvAlloc*       alloc);
```

PARAMETERS

phone

A pointer to a valid *RvSdpPhone*.

badSyn

The proprietary format of the *RvSdpPhone*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpPhone*, or NULL if the function fails.

rvSdpPhoneIsBadSyntax()

DESCRIPTION

Indicates whether or not the *RvSdpPhone* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpPhoneIsBadSyntax(
    RvSdpPhone*    phone);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax or RV_FALSE otherwise.

rvSdpPhoneDestruct()

DESCRIPTION

Destructs an *RvSdpPhone*.

SYNTAX

```
void rvSdpPhoneDestruct(
    RvSdpPhone*    phone);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

RETURN VALUES

None.

rvSdpPhoneCopy()

DESCRIPTION

Copies the values from a source *RvSdpPhone* to a destination *RvSdpPhone*.

SYNTAX

```
RvSdpPhone* rvSdpPhoneCopy(
    RvSdpPhone*          dest,
    const RvSdpPhone*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpPhone*. This parameter must point to a constructed *RvSdpPhone*.

src

The source *RvSdpPhone*.

RETURN VALUES

Returns a pointer to the destination *RvSdpPhone*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpPhoneGetBadSyntax()
- rvSdpPhoneSetBadSyntax()
- rvSdpPhoneGetNumber()
- rvSdpPhoneSetNumber()
- rvSdpPhoneGetText()
- rvSdpPhoneSetText()

rvSdpPhoneGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpPhone* field value or empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpPhoneGetBadSyntax(
    const RvSdpPhone*    phone);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

RETURN VALUES

Returns the bad syntax value.

rvSdpPhoneSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpPhone* field value to the proprietary format.

SYNTAX

```
RvSdpStatus rvSdpPhoneSetBadSyntax(
    RvSdpPhone*    o,
    const char*    bs);
```

PARAMETERS

o

A pointer to the *RvSdpPhone*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPhoneGetNumber()

DESCRIPTION

Gets the *RvSdpPhone* number.

SYNTAX

```
const char* rvSdpPhoneGetNumber(
    const RvSdpPhone*    phone);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

RETURN VALUES

Returns the *RvSdpPhone* number.

rvSdpPhoneSetNumber()

DESCRIPTION

Sets the *RvSdpPhone* number.

SYNTAX

```
RvSdpStatus rvSdpPhoneSetNumber(
    RvSdpPhone*    phone,
    const char*    number);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

number

The *RvSdpPhone* number.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPhoneGetText()

DESCRIPTION

Gets the *RvSdpPhone* text.

SYNTAX

```
const char* rvSdpPhoneGetText(
    const RvSdpPhone*    phone);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

RETURN VALUES

Returns the *RvSdpPhone* text.

rvSdpPhoneSetText()

DESCRIPTION

Sets the *RvSdpPhone* text.

SYNTAX

```
RvSdpStatus rvSdpPhoneSetText(
    RvSdpPhone*    phone,
    const char*    text);
```

PARAMETERS

phone

A pointer to the *RvSdpPhone*.

text

The *RvSdpPhone* text.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

10

CONNECTION FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used for operating on *RvSdpConnection* objects. The RvSdpConnection Type represents the connection ('c=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpConnectionConstruct() **Obsolete**
- rvSdpConnectionConstructA() **Obsolete**
- rvSdpConnectionConstructCopy() **Obsolete**
- rvSdpConnectionConstructCopyA() **Obsolete**
- rvSdpBadSyntaxConnectionConstruct() **Obsolete**
- rvSdpBadSyntaxConnectionConstructA **Obsolete**
- rvSdpConnectionIsBadSyntax()
- rvSdpConnectionDestruct()
- rvSdpConnectionCopy()

rvSdpConnectionConstruct()

DESCRIPTION

Constructs an *RvSdpConnection*.

This function is obsolete. Please use rvSdpMsgAddConnection() or rvSdpMediaDescrAddConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpConnectionConstruct(
    RvSdpConnection*    conn,
    RvSdpNetType        nettype,
    RvSdpAddrType       addrtype,
    const char*         address);
```

PARAMETERS

conn

A pointer to a valid *RvSdpConnection*.

nettype

The network type.

addrtype

The address type.

addr

The *RvSdpConnection* address.

RETURN VALUES

Returns a pointer to the constructed *RvSdpConnection*, or NULL if the function fails.

rvSdpConnectionConstructA()

DESCRIPTION

Constructs an *RvSdpConnection*.

This function is obsolete. Please use rvSdpMsgAddConnection() or rvSdpMediaDescrAddConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpConnectionConstructA(
    RvSdpConnection*    conn,
    RvSdpNetType        nettype,
    RvSdpAddrType       addrtype,
    const char*         address,
    RvAlloc*            a);
```

PARAMETERS

conn

A pointer to a valid *RvSdpConnection*.

nettype

The network type.

addrtype

The address type.

addr

The *RvSdpConnection* address.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpConnection*, or NULL if the function fails.

rvSdpConnectionConstructCopy()

DESCRIPTION

Constructs an *RvSdpConnection* and copies the values from a source *RvSdpConnection* field.

This function is obsolete. Please use rvSdpMsgAddConnection() or rvSdpMediaDescrAddConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpConnectionConstructCopy(
    RvSdpConnection*          dest,
    const RvSdpConnection*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpConnection* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpConnection*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpConnection*, or NULL if the function fails.

rvSdpConnectionConstructCopyA()

DESCRIPTION

Constructs an *RvSdpConnection* and copies the values from a source *RvSdpConnection* field.

This function is obsolete. Please use rvSdpMsgAddConnection() or rvSdpMediaDescrAddConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpConnectionConstructCopyA(
    RvSdpConnection*          dest,
    const RvSdpConnection*    src,
    RvAlloc*                  a);
```

PARAMETERS

dest

A pointer to the *RvSdpConnection* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpConnection*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the *RvSdpConnection*, or NULL if the function fails.

rvSdpBadSyntaxConnectionConstruct()

DESCRIPTION

Constructs an *RvSdpConnection* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxConnection() or rvSdpMediaDescrSetBadSyntaxConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpBadSyntax ConnectionConstruct(
    RvSdpConnection*    conn,
    const char*         badSyn);
```

PARAMETERS

conn

A pointer to a valid *RvSdpConnection*.

badSyn

The proprietary format of the *RvSdpConnection*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpConnection*, or NULL if the function fails.

rvSdpBadSyntaxConnectionConstructA

DESCRIPTION

Constructs an *RvSdpConnection* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxConnection() or rvSdpMediaDescrSetBadSyntaxConnection() instead.

SYNTAX

```
RvSdpConnection* rvSdpBadSyntax ConnectionConstructA(
    RvSdpConnection*    conn,
    const char*         badSyn,
    RvAlloc*            a);
```

PARAMETERS

conn

A pointer a valid *RvSdpConnection*.

badSyn

The proprietary format of the *RvSdpConnection*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpConnection*, or NULL if the function fails.

rvSdpConnectionIsBadSyntax()

DESCRIPTION

Indicates whether or not the *RvSdpConnection* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpConnectionIsBadSyntax(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpConnectionDestruct()

DESCRIPTION

Destructs an *RvSdpConnection*.

SYNTAX

```
void rvSdpConnectionDestruct(
    RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

None.

rvSdpConnectionCopy()

DESCRIPTION

Copies the values from a source *RvSdpConnection* to a destination *RvSdpConnection*.

SYNTAX

```
RvSdpConnection* rvSdpConnectionCopy(
    RvSdpConnection*          dest,
    const RvSdpConnection*    src);
```

PARAMETERS

dest

A pointer to a destination *RvSdpConnection*. This parameter must point to a constructed *RvSdpConnection*.

src

The source *RvSdpConnection*.

RETURN VALUES

Returns a pointer to the destination *RvSdpConnection*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpConnectionGetBadSyntax()
- rvSdpConnectionSetBadSyntax()
- rvSdpConnectionGetNetType()
- rvSdpConnectionSetNetType()
- rvSdpConnectionGetNetTypeStr()
- rvSdpConnectionSetNetTypeStr()
- rvSdpConnectionGetAddrType()
- rvSdpConnectionSetAddrType()
- rvSdpConnectionGetAddrTypeStr()
- rvSdpConnectionSetAddrTypeStr()
- rvSdpConnectionGetAddress()
- rvSdpConnectionSetAddress()
- rvSdpConnectionGetAddressTTL()
- rvSdpConnectionSetAddressTTL()
- rvSdpConnectionGetAddressNum()
- rvSdpConnectionSetAddressNum()

rvSdpConnectionGetBadSyntax()

DESCRIPTION

Gets a proprietary formatted *RvSdpConnection* field value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpConnectionGetBadSyntax(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the bad syntax value.

rvSdpConnectionSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpConnection* field value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpConnectionSetBadSyntax(
    RvSdpConnection*    o,
    const char*         bs);
```

PARAMETERS

o

A pointer to the *RvSdpConnection*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpConnectionGetNetType()

DESCRIPTION

Gets the network type of an *RvSdpConnection*.

SYNTAX

```
RvSdpNetType rvSdpConnectionGetNetType(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the network type of the *RvSdpConnection*.

rvSdpConnectionSetNetType()

DESCRIPTION

Sets the network type of an *RvSdpConnection*.

SYNTAX

```
void rvSdpConnectionSetNetType(
    RvSdpConnection*    conn,
    RvSdpNetType        type);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

type

The network type.

RETURN VALUES

None.

rvSdpConnectionGetNetTypeStr()

DESCRIPTION

Gets the network type string of an *RvSdpConnection*.

SYNTAX

```
const char* rvSdpConnectionGetNetTypeStr(
    RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the network type string of the *RvSdpConnection*.

rvSdpConnectionSetNetTypeStr()

DESCRIPTION

Sets the network type string of the *RvSdpConnection*.

SYNTAX

```
RvSdpStatus rvSdpConnectionSetNetTypeStr(
    RvSdpConnection*    conn,
    const char*         type);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

type

The network type.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpConnectionGetAddrType()

DESCRIPTION

Gets the address type of an *RvSdpConnection*.

SYNTAX

```
RvSdpAddrType rvSdpConnectionGetAddrType(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the address type of the *RvSdpConnection*.

rvSdpConnectionSetAddrType()

DESCRIPTION

Sets the address type of an *RvSdpConnection*.

SYNTAX

```
void rvSdpConnectionSetAddrType(
    RvSdpConnection*    conn,
    RvSdpAddrType       type);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

type

The address type.

RETURN VALUES

None.

rvSdpConnectionGetAddrTypeStr()

DESCRIPTION

Gets the address type string of an *RvSdpConnection*.

SYNTAX

```
const char* rvSdpConnectionGetAddrTypeStr(
    RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the address type string of the *RvSdpConnection*.

rvSdpConnectionSetAddrTypeStr()

DESCRIPTION

Sets the address type string of an *RvSdpConnection*.

SYNTAX

```
RvSdpStatus rvSdpConnectionSetAddrTypeStr(
    RvSdpConnection*    conn,
    const char*         type);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

type

The address type string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpConnectionGetAddress()

DESCRIPTION

Gets the address of an *RvSdpConnection*.

SYNTAX

```
const char* rvSdpConnectionGetAddress(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the address of the *RvSdpConnection*.

rvSdpConnectionSetAddress()

DESCRIPTION

Sets the address of an *RvSdpConnection*.

SYNTAX

```
RvSdpStatus rvSdpConnectionSetAddress(
    RvSdpConnection*    conn,
    const char*         addr);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

addr

The address of the *RvSdpConnection*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpConnectionGetAddressTTL()

DESCRIPTION

Gets the address TTL of an *RvSdpConnection*.

SYNTAX

```
int rvSdpConnectionGetAddressTTL(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the address TTL of the *RvSdpConnection*.

rvSdpConnectionSetAddressTTL()

DESCRIPTION

Sets the address TTL of an *RvSdpConnection*.

SYNTAX

```
void rvSdpConnectionSetAddressTTL(
    RvSdpConnection*    conn,
    int ttl);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

ttl

The address TTL of the *RvSdpConnection*.

RETURN VALUES

None.

rvSdpConnectionGetAddressNum()

DESCRIPTION

Gets the number of subsequent addresses of an *RvSdpConnection*.

SYNTAX

```
int rvSdpConnectionGetAddressNum(
    const RvSdpConnection*    conn);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

RETURN VALUES

Returns the number of subsequent addresses of the *RvSdpConnection*.

rvSdpConnectionSetAddressNum()

DESCRIPTION

Sets the number of subsequent addresses of an *RvSdpConnection*.

SYNTAX

```
void rvSdpConnectionSetAddressNum(
    RvSdpConnection*    conn,
    nt                  num);
```

PARAMETERS

conn

A pointer to the *RvSdpConnection*.

num

The number of subsequent addresses of the *RvSdpConnection*.

RETURN VALUES

None.

11

BANDWIDTH FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpBandwidth* objects. The *RvSdpBandwidth* Type represents the bandwidth ('b=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpBandwidthConstruct() **Obsolete**
- rvSdpBandwidthConstructA() **Obsolete**
- rvSdpBandwidthConstructCopy() **Obsolete**
- rvSdpBandwidthConstructCopyA() **Obsolete**
- rvSdpBadSyntaxBandwidthConstruct() **Obsolete**
- rvSdpBadSyntaxBandwidthConstructA() **Obsolete**
- rvSdpBandwidthIsBadSyntax()
- rvSdpBandwidthDestruct()
- rvSdpBandwidthCopy()

rvSdpBandwidthConstruct()

DESCRIPTION

Constructs an *RvSdpBandwidth*.

This function is obsolete. Please use rvSdpMsgAddBandwidth() or rvSdpMediaDescrAddBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthConstruct(
    RvSdpBandwidth*    bw,
    const char*        type,
    RvUint32           value);
```

PARAMETERS

bw

A pointer to a valid *RvSdpBandwidth*.

type

The type name of an *RvSdpBandwidth*.

value

The value (in Kbs) of the *RvSdpBandwidth*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpBandwidth*, or NULL if the function fails.

rvSdpBandwidthConstructA()

DESCRIPTION

Constructs an *RvSdpBandwidth*.

This function is obsolete. Please use rvSdpMsgAddBandwidth() or rvSdpMediaDescrAddBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthConstructA(
    RvSdpBandwidth*    bw,
    const char*        type,
    RvUint32           value,
    RvAlloc*           a);
```

PARAMETERS

bw

A pointer to a valid *RvSdpBandwidth*.

type

The type name of the *RvSdpBandwidth*.

value

The value (in Kbs) of the of the *RvSdpBandwidth*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpBandwidth*, or NULL if the function fails.

rvSdpBandwidthConstructCopy()

DESCRIPTION

Constructs an *RvSdpBandwidth* and copies the values from a source *RvSdpBandwidth* field.

This function is obsolete. Please use rvSdpMsgAddBandwidth() or rvSdpMediaDescrAddBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthConstructCopy(
    RvSdpBandwidth*    dest,
    RvSdpBandwidth*    src);
```

PARAMETERS

dest

A pointer to an *RvSdpBandwidth* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpBandwidth*.

RETURN VALUES

Returns the pointer to the constructed object, or NULL if the function fails.

rvSdpBandwidthConstructCopyA()

DESCRIPTION

Constructs an *RvSdpBandwidth* and copies the values from a source *RvSdpBandwidth* field.

This function is obsolete. Please use rvSdpMsgAddBandwidth() or rvSdpMediaDescrAddBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthConstructCopyA(
    RvSdpBandwidth*    dest,
    RvSdpBandwidth*    src,
    RvAlloc*           a);
```

PARAMETERS

dest

A pointer to the *RvSdpBandwidth* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpBandwidth*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpBandwidth*, or NULL if the function fails.

rvSdpBadSyntaxBandwidthConstruct()

DESCRIPTION

Constructs an *RvSdpBandwidth* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxBandwidth() or rvSdpMediaDescrSetBadSyntaxBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBadSyntax BandwidthConstruct(
    RvSdpBandwidth*    bw,
    const char*        badSyn);
```

PARAMETERS

bw

A pointer to a valid *RvSdpBandwidth*.

badSyn

The proprietary format of the *RvSdpBandwidth* field.

RETURN VALUES

Returns a pointer to the constructed *RvSdpBandwidth*, or NULL if the function fails.

rvSdpBadSyntaxBandwidthConstructA()

DESCRIPTION

Constructs an *RvSdpBandwidth* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxBandwidth() or rvSdpMediaDescrSetBadSyntaxBandwidth() instead.

SYNTAX

```
RvSdpBandwidth* rvSdpBadSyntaxBandwidthConstructA(
    RvSdpBandwidth*    bw,
    const char*        badSyn,
    RvAlloc*           a);
```

PARAMETERS

bw

A pointer to a valid *RvSdpBandwidth*.

badSyn

The proprietary format of the *RvSdpBandwidth* field.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpBandwidth*, or NULL if the function fails.

rvSdpBandwidthIsBadSyntax()

DESCRIPTION

Indicates whether or not the *RvSdpBandwidth* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpBandwidthIsBadSyntax(
    RvSdpBandwidth*    bw);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpBandwidthDestruct()

DESCRIPTION

Destructs an *RvSdpBandwidth*.

SYNTAX

```
void rvSdpBandwidthDestruct(
    RvSdpBandwidth*    bw);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

RETURN VALUES

None.

rvSdpBandwidthCopy()

DESCRIPTION

Copies the values from a source *RvSdpBandwidth* to a destination *RvSdpBandwidth*.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthCopy(
    RvSdpBandwidth*    dest,
    RvSdpBandwidth*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpBandwidth*. This parameter must point to the constructed *RvSdpBandwidth*.

src

A source *RvSdpBandwidth*.

RETURN VALUES

Returns a pointer to the destination *RvSdpBandwidth*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpBandwidthGetBadSyntax()
- rvSdpBandwidthSetBadSyntax()
- rvSdpBandwidthGetType()
- rvSdpBandwidthSetType()
- rvSdpBandwidthGetValue()
- rvSdpBandwidthSetValue()

rvSdpBandwidthGetBadSyntax()

DESCRIPTION

Gets a proprietary formatted *RvSdpBandwidth* field value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpBandwidthGetBadSyntax(
    const RvSdpBandwidth*    bw);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

RETURN VALUES

Returns the bad syntax value.

rvSdpBandwidthSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpBandwidth* field value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpBandwidthSetBadSyntax(
    RvSdpBandwidth*    o,
    const char*        bs);
```

PARAMETERS

o

A pointer to the *RvSdpBandwidth*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpBandwidthGetType()

DESCRIPTION

Gets the type of *RvSdpBandwidth*.

SYNTAX

```
const char* rvSdpBandwidthGetType(
    const RvSdpBandwidth*    bw);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

RETURN VALUES

Returns the type of *RvSdpBandwidth*.

rvSdpBandwidthSetType()

DESCRIPTION

Sets the type of *RvSdpBandwidth*.

SYNTAX

```
RvSdpStatus rvSdpBandwidthSetType(
    RvSdpBandwidth*    bw,
    const char*        type);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

type

The type of *RvSdpBandwidth*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpBandwidthGetValue()

DESCRIPTION

Gets the value of an *RvSdpBandwidth*.

SYNTAX

```
RvUint32 rvSdpBandwidthGetValue(
    const RvSdpBandwidth*    bw);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

RETURN VALUES

Returns the value of the *RvSdpBandwidth*.

rvSdpBandwidthSetValue()

DESCRIPTION

Sets the value of an *RvSdpBandwidth*.

SYNTAX

```
void rvSdpBandwidthSetValue(
    RvSdpBandwidth*    bw,
    RvUint32           value);
```

PARAMETERS

bw

A pointer to the *RvSdpBandwidth*.

value

The value of the *RvSdpBandwidth*.

RETURN VALUES

None.

12

SESSION TIME FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpSessionTime* objects. The *RvSdpSessionTime* Type represents the time ('t=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpSessionTimeConstruct() **Obsolete**
- rvSdpSessionTimeConstructA() **Obsolete**
- rvSdpSessionTimeConstructCopy() **Obsolete**
- rvSdpSessionTimeConstructCopyA() **Obsolete**
- rvSdpBadSyntaxSessionTimeConstruct() **Obsolete**
- rvSdpBadSyntaxSessionTimeConstructA() **Obsolete**
- rvSdpSessionTimeIsBadSyntax()
- rvSdpSessionTimeDestruct()
- rvSdpSessionTimeCopy()
- rvSdpSessionTimeAddRepeatInterval()
- rvSdpSessionTimeAddBadSyntaxRepeatInterval()
- rvSdpSessionTimeRemoveCurrentRepeatInterval()
- rvSdpSessionTimeRemoveRepeatInterval()
- rvSdpSessionTimeClearRepeatIntervals()

rvSdpSessionTimeConstruct()

DESCRIPTION

Constructs an *RvSdpSessionTime*.

This function is obsolete. Please use rvSdpMsgAddSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeConstruct(
    RvSdpSessionTime*    sessTime,
    RvUint32             start,
    RvUint32             end);
```

PARAMETERS

sessTime

A pointer to valid *RvSdpSessionTime*.

start

The start time of the session.

end

The end time of the session.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpSessionTimeConstructA()

DESCRIPTION

Constructs an *RvSdpSessionTime*.

This function is obsolete. Please use rvSdpMsgAddSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeConstructA(
    RvSdpSessionTime*    sessTime,
    RvUint32             start,
    RvUint32             end,
    RvAlloc*             a);
```

PARAMETERS

sessTime

A pointer to a valid *RvSdpSessionTime*.

start

The start time of the session.

end

The end time of the session.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpSessionTimeConstructCopy()

DESCRIPTION

Constructs an *RvSdpSessionTime* and copies the values from a source *RvSdpSessionTime*.

This function is obsolete. Please use rvSdpMsgAddSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeConstructCopy(
    RvSdpSessionTime*          dest,
    const RvSdpSessionTime*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpSessionTime* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpSessionTime*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpSessionTimeConstructCopyA()

DESCRIPTION

Constructs an *RvSdpSessionTime* and copies the values from a source *RvSdpSessionTime*.

This function is obsolete. Please use rvSdpMsgAddSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeConstructCopyA(
    RvSdpSessionTime*          dest,
    const RvSdpSessionTime*    src,
    RvAlloc*                   a);
```

PARAMETERS

dest

A pointer to the *RvSdpSessionTime* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpSessionTime*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpBadSyntaxSessionTimeConstruct()

DESCRIPTION

Constructs an *RvSdpSessionTime* with proprietary format using the default *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpBadSyntaxSessionTimeConstruct(
    RvSdpSessionTime*    sessTime,
    const char*          badSyn);
```

PARAMETERS

descr

A pointer to a valid *RvSdpSessionTime*.

badSyn

The proprietary format of the *RvSdpSessionTime*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpBadSyntaxSessionTimeConstructA()

DESCRIPTION

Constructs an *RvSdpSessionTime* with proprietary format using the provided *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxSessionTime() instead.

SYNTAX

```
RvSdpSessionTime* rvSdpBadSyntaxSessionTimeConstructA(
    RvSdpSessionTime*    sessTime,
    const char*          badSyn,
    RvAlloc*             a);
```

PARAMETERS

descr

A pointer to a valid *RvSdpSessionTime*.

badSyn

The proprietary format of the *RvSdpSessionTime*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpSessionTime*, or NULL if the function fails.

rvSdpSessionTimeIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpSessionTime* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpSessionTimeIsBadSyntax(
    RvSdpSessionTime*    sessTime);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpSessionTimeDestruct()

DESCRIPTION

Destructs an *RvSdpSessionTime*.

SYNTAX

```
void rvSdpSessionTimeDestruct(
    RvSdpSessionTime*    sessTime);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

RETURN VALUES

None.

rvSdpSessionTimeCopy()

DESCRIPTION

Copies the values from a source *RvSdpSessionTime* to a destination *RvSdpSessionTime*.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeCopy(
    RvSdpSessionTime*          dest,
    const RvSdpSessionTime*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpSessionTime*. This parameter must point to a constructed *RvSdpSessionTime*.

src

The *RvSdpSessionTime*.

RETURN VALUES

Returns a pointer to the destination *RvSdpSessionTime*, or NULL if the function fails.

rvSdpSessionTimeAddRepeatInterval()

DESCRIPTION

Adds a new *RvSdpRepeatInterval* to an *RvSdpSessionTime*

SYNTAX

```
RvSdpRepeatInterval* rvSdpSessionTimeAddRepeatInterval(
    RvSdpSessionTime*    session,
    RvUint32             time,
    RvSdpTimeUnit        t_units,
    RvUint32             duration,
    RvSdpTimeUnit        d_units);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

time

The length of the time interval of the *RvSdpRepeatInterval*.

t_units

The time units of the *RvSdpRepeatInterval*.

duration

The length of the active duration.

d_units

The time units of the active duration.

RETURN VALUES

Returns the pointer to added *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpSessionTimeAddBadSyntaxRepeatInterval()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpRepeatInterval* to an *RvSdpSessionTime*.

SYNTAX

```
RvSdpRepeatInterval*
  rvSdpSessionTimeAddBadSyntaxRepeatInterval(
    RvSdpSessionTime*    session,
    const char*          badSyn);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

badSyn

The proprietary value of the *RvSdpSessionTime*.

RETURN VALUES

Returns a pointer to the newly created *RvSdpRepeatInterval* if the function succeeds, or a NULL pointer if the function fails.

rvSdpSessionTimeRemoveCurrentRepeatInterval()

DESCRIPTION

Removes (and destructs) an *RvSdpRepeatInterval* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpSessionTimeRemoveCurrentRepeatInterval(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpSessionTimeGetFirstRepeatInterval() or rvSdpSessionTimeGetNextRepeatInterval() function.

RETURN VALUES

None.

rvSdpSessionTimeRemoveRepeatInterval()

DESCRIPTION

Removes (and destructs) an *RvSdpRepeatInterval* by index in the context of an *RvSdpSessionTime*.

SYNTAX

```
void rvSdpSessionTimeRemoveRepeatInterval(
    RvSdpSessionTime*    session,
    RvSize_t             index);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpSessionTimeGetNumOfRepeatInterval().

RETURN VALUES

None.

rvSdpSessionTimeClearRepeatIntervals()

DESCRIPTION

Removes (and destructs) all *RvSdpRepeatInterval* objects set in an *RvSdpSessionTime*.

SYNTAX

```
void rvSdpSessionTimeClearRepeatIntervals(
    RvSdpSessionTime*    session);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpSessionTimeGetBadSyntax()
- rvSdpSessionTimeSetBadSyntax()
- rvSdpSessionTimeGetStart()
- rvSdpSessionTimeSetStart()
- rvSdpSessionTimeGetEnd()
- rvSdpSessionTimeSetEnd()
- rvSdpSessionTimeGetNumOfRepeatInterval()
- rvSdpSessionTimeGetFirstRepeatInterval()
- rvSdpSessionTimeGetNextRepeatInterval()
- rvSdpSessionTimeGetRepeatInterval()

rvSdpSessionTimeGetBadSyntax()

DESCRIPTION

Gets a proprietary formatted *RvSdpSessionTime* field value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpSessionTimeGetBadSyntax(
    const RvSdpSessionTime*    sessTime);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

RETURN VALUES

Returns the bad syntax value.

rvSdpSessionTimeSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpSessionTime* field value to proprietary-format.

SYNTAX

```
RvSdpStatus rvSdpSessionTimeSetBadSyntax(
    RvSdpSessionTime*    o,
    const char*          bs);
```

PARAMETERS

o

A pointer to the *RvSdpSessionTime*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpSessionTimeGetStart()

DESCRIPTION

Gets the start time of an *RvSdpSessionTime*.

SYNTAX

```
RvUint32 rvSdpSessionTimeGetStart(
    RvSdpSessionTime*    sessTime);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

RETURN VALUES

Returns the session start time.

rvSdpSessionTimeSetStart()

DESCRIPTION

Sets the start time of an *RvSdpSessionTime*.

SYNTAX

```
void rvSdpSessionTimeSetStart(
    RvSdpSessionTime*    sessTime,
    RvUint32             start);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

start

The new start time of the *RvSdpSessionTime*.

RETURN VALUES

None.

rvSdpSessionTimeGetEnd()

DESCRIPTION

Gets the end time of an *RvSdpSessionTime*.

SYNTAX

```
RvUint32 rvSdpSessionTimeGetEnd(
    RvSdpSessionTime*    sessTime);
```

PARAMETERS

sessTime

A pointer to *RvSdpSessionTime*.

RETURN VALUES

Returns the end time of the *RvSdpSessionTime*.

rvSdpSessionTimeSetEnd()

DESCRIPTION

Sets the end time of an *RvSdpSessionTime*.

SYNTAX

```
void rvSdpSessionTimeSetEnd(
    RvSdpSessionTime*    sessTime,
    RvUint32             end);
```

PARAMETERS

sessTime

A pointer to the *RvSdpSessionTime*.

end

The new end time of the *RvSdpSessionTime*.

RETURN VALUES

None.

rvSdpSessionTimeGetNumOfRepeatInterval()

DESCRIPTION

Gets the number of *RvSdpRepeatInterval* objects of an *RvSdpSessionTime*.

SYNTAX

```
RvSize_t rvSdpSessionTimeGetNumOfRepeatInterval(
    const RvSdpSessionTime*    session);
```

PARAMETERS

session

A pointer to the RvSdpSessionTime.

RETURN VALUES

Returns the number of defined *RvSdpRepeatInterval* objects.

rvSdpSessionTimeGetFirstRepeatInterval()

DESCRIPTION

Returns the first *RvSdpRepeatInterval* defined in an *RvSdpSessionTime*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpRepeatInterval* rvSdpSessionTimeGetFirstRepeatInterval(
    RvSdpSessionTime*    session,
    RvSdpListIter*       iter);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

iter

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpSessionTimeGetNextRepeatInterval() calls.

RETURN VALUES

Returns a pointer to the *RvSdpRepeatInterval*, or a NULL pointer if there are no repeat intervals defined in the *RvSdpSessionTime*.

rvSdpSessionTimeGetNextRepeatInterval()

DESCRIPTION

Returns the next *RvSdpRepeatInterval* defined in an *RvSdpSessionTime*. The next object is defined based on the state of the *RvSdpListIter.*

SYNTAX

```
RvSdpRepeatInterval* rvSdpSessionTimeGetNextRepeatInterval(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpSessionTimeGetFirstRepeatInterval() or rvSdpSessionTimeGetNextRepeatInterval() function.

RETURN VALUES

Returns a pointer to the *RvSdpRepeatInterval*, or a NULL pointer if there are no more repeat intervals defined in the *RvSdpSessionTime*.

rvSdpSessionTimeGetRepeatInterval()

DESCRIPTION

Gets an *RvSdpRepeatInterval* by index (in the in the context of an *RvSdpSessionTime*.

SYNTAX

```
RvSdpRepeatInterval* rvSdpSessionTimeGetRepeatInterval(
    const RvSdpSessionTime*    session,
    RvSize_t                   i);
```

PARAMETERS

session

A pointer to the *RvSdpSessionTime*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpSessionTimeGetNumOfRepeatInterval().

RETURN VALUES

Returns the requested *RvSdpRepeatInterval* pointer.

13

TIME REPEAT INTERVAL FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpRepeatInterval* objects. The RvSdpRepeatInterval Type represents the repeat interval ('r=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpRepeatIntervalConstruct() **Obsolete**
- rvSdpRepeatIntervalConstructA() **Obsolete**
- rvSdpRepeatIntervalConstructCopy() **Obsolete**
- rvSdpBadSyntaxRepeatIntervalConstructA() **Obsolete**
- rvSdpBadSyntaxRepeatIntervalConstruct() **Obsolete**
- rvSdpRepeatIntervalIsBadSyntax()
- rvSdpRepeatIntervalDestruct()
- rvSdpRepeatIntervalCopy()
- rvSdpRepeatIntervalAddOffset()
- rvSdpRepeatIntervalRemoveCurrentOffset()
- rvSdpRepeatIntervalRemoveOffset()
- rvSdpRepeatIntervalClearOffset()

rvSdpRepeatIntervalConstruct()

DESCRIPTION

Constructs an *RvSdpRepeatInterval*.

This function is obsolete. Please use rvSdpSessionTimeAddRepeatInterval() instead.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalConstruct(
    RvSdpRepeatInterval*    interv,
    RvUint32                time,
    RvSdpTimeUnit           t_units,
    RvUint32                duration,
    RvSdpTimeUnit           d_units);
```

PARAMETERS

interv

A pointer to a valid *RvSdpRepeatInterval*.

time

The length of the time interval of the *RvSdpRepeatInterval*.

t_units

The time units of the *RvSdpRepeatInterval*.

duration

The length of the active duration.

d_units

The time units of the active duration.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpRepeatIntervalConstructA()

DESCRIPTION

Constructs an *RvSdpRepeatInterval*.

This function is obsolete. Please use rvSdpSessionTimeAddRepeatInterval() instead.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalConstructA(
    RvSdpRepeatInterval*    interv,
    RvUint32                time,
    RvSdpTimeUnit           t_units,
    RvUint32                duration,
    RvSdpTimeUnit           d_units,
    RvAlloc*                a);
```

PARAMETERS

interv

A pointer to a valid *RvSdpRepeatInterval*.

time

The length of the time interval of the *RvSdpRepeatInterval*.

t_units

The time units of the *RvSdpRepeatInterval*.

duration

The length of the active duration.

d_units

The time units of the active duration.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpRepeatIntervalConstructCopy()

DESCRIPTION

Constructs an *RvSdpRepeatInterval* and copies the values from a source *RvSdpRepeatInterval*.

This function is obsolete. Please use rvSdpSessionTimeAddRepeatInterval() instead.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalConstructCopy(
    RvSdpRepeatInterval          *d,
    const RvSdpRepeatInterval    *s,
    RvAlloc*                     alloc);
```

PARAMETERS

dest

A pointer to the *RvSdpRepeatInterval* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpRepeatInterval*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpBadSyntaxRepeatIntervalConstructA()

DESCRIPTION

Constructs an *RvSdpRepeatInterval* with proprietary format.

This function is obsolete. Please use rvSdpSessionTimeAddRepeatInterval() instead.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalBadSyntaxConstructA(
    RvSdpRepeatInterval*    interv,
    char                    *badSyntax,
    RvAlloc*                a);
```

PARAMETERS

descr

A pointer to a valid *RvSdpRepeatInterval*.

badSyn

The proprietary format of the *RvSdpRepeatInterval*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpBadSyntaxRepeatIntervalConstruct()

DESCRIPTION

Constructs an *RvSdpRepeatInterval* with proprietary format.

This function is obsolete. Please use rvSdpSessionTimeAddBadSyntaxRepeatInterval() instead.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalBadSyntaxConstruct(
    RvSdpRepeatInterval*    interv,
    char                    *badSyntax);
```

PARAMETERS

descr

A pointer to a valid *RvSdpRepeatInterval*.

badSyn

The proprietary format of the *RvSdpRepeatInterval*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpRepeatIntervalIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpRepeatInterval* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpRepeatIntervalIsBadSyntax(
    RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpRepeatIntervalDestruct()

DESCRIPTION

Destructs an *RvSdpRepeatInterval*.

SYNTAX

```
void rvSdpRepeatIntervalDestruct(
    RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

None.

rvSdpRepeatIntervalCopy()

DESCRIPTION

Copies the values from a source *RvSdpRepeatInterval* to a destination. *RvSdpRepeatInterval*.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntervalCopy(
    RvSdpRepeatInterval          *d,
    const RvSdpRepeatInterval    *s);
```

PARAMETERS

dest

A pointer to the destination *RvSdpRepeatInterval*. This parameter must point to the constructed *RvSdpRepeatInterval*.

src

The *RvSdpRepeatInterval*.

RETURN VALUES

Returns a pointer to the destination *RvSdpRepeatInterval*, or NULL if the function fails.

rvSdpRepeatIntervalAddOffset()

DESCRIPTION

Adds another session start offset time to an *RvSdpRepeatInterval*.

SYNTAX

```
RvSdpStatus rvSdpRepeatIntervalAddOffset(
    RvSdpRepeatInterval*    repeat,
    RvUint32                time,
    RvSdpTimeUnit           units);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

time

The session start offset time length.

units

The session start offset time units.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpRepeatIntervalRemoveCurrentOffset()

DESCRIPTION

Removes (and destructs) an *RvSdpRepeatInterval* to which the *iter* parameter points. The value of *iter* is undefined after the function call.

SYNTAX

```
void rvSdpRepeatIntervalRemoveCurrentOffset(
    RvSdpListIter*    iter);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpRepeatIntervalGetFirstOffset() or rvSdpRepeatIntervalGetNextOffset() function.

RETURN VALUES

None.

rvSdpRepeatIntervalRemoveOffset()

DESCRIPTION

Removes (and destructs) an *RvSdpRepeatInterval* offset by index.

SYNTAX

```
void rvSdpRepeatIntervalRemoveOffset(
    RvSdpRepeatInterval*    repeat,
    RvSize_t                index);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpRepeatIntervalGetNumOfOffset() call.

RETURN VALUES

None.

rvSdpRepeatIntervalClearOffset()

DESCRIPTION

Removes (and destructs) all offsets set in an *RvSdpRepeatInterval*.

SYNTAX

```
void rvSdpRepeatIntervalClearOffset(
    RvSdpRepeatInterval*    repeat);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpRepeatIntervalGetBadSyntax()
- rvSdpRepeatIntervalSetBadSyntax()
- rvSdpRepeatIntervalGetNumOfOffset()
- rvSdpRepeatIntervalGetFirstOffset()
- rvSdpRepeatIntervalGetNextOffset()
- rvSdpRepeatIntervalGetOffsetTime()
- rvSdpRepeatIntervalGetOffsetUnits()
- rvSdpRepeatIntervalGetDurationUnits()
- rvSdpRepeatIntervalSetDurationUnits()
- rvSdpRepeatIntervalGetDurationTime()
- rvSdpRepeatIntervalSetDurationTime()
- rvSdpRepeatIntervalGetIntervalUnits()
- rvSdpRepeatIntervalSetIntervalUnits()
- rvSdpRepeatIntervalGetIntervalTime()
- rvSdpRepeatIntervalSetIntervalTime()

rvSdpRepeatIntervalGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpRepeatInterval* field value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpRepeatIntervalGetBadSyntax(
    RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the bad syntax value.

rvSdpRepeatIntervalSetBadSyntax()

DESCRIPTION

Sets an *RvSdpRepeatInterval* field value to proprietary-formatted.

SYNTAX

```
RvSdpStatus rvSdpRepeatIntervalSetBadSyntax(
    RvSdpRepeatInterval*    o,
    const char*             bs);
```

PARAMETERS

o

A pointer to the *RvSdpRepeatInterval*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpRepeatIntervalGetNumOfOffset()

DESCRIPTION

Gets the number of offsets in an *RvSdpRepeatInterval*.

SYNTAX

```
RvSize_t rvSdpRepeatIntervalGetNumOfOffset(
    RvSdpRepeatInterval*    repeat);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the number of defined offsets.

rvSdpRepeatIntervalGetFirstOffset()

DESCRIPTION

Returns the first offset data defined in an *RvSdpRepeatInterval*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvBool rvSdpRepeatIntervalGetFirstOffset(
    RvSdpRepeatInterval*    repeat,
    RvSdpListIter*          i,
    RvUint32*               time,
    RvSdpTimeUnit*          t_unit);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

i

A pointer to the *RvSdpListIter* to be used for subsequent rvSdpRepeatIntervalGetNextOffset() calls.

time

The offset time interval.

t_unit

The offset time units.

RETURN VALUES

Returns RV_TRUE if there is at least one offset in the *RvSdpRepeatInterval*.

rvSdpRepeatIntervalGetNextOffset()

DESCRIPTION

Returns the next offset data defined in an *RvSdpRepeatInterval*. The next object is defined based on the state of an *RvSdpListIter*.

SYNTAX

```
RvBool rvSdpRepeatIntervalGetNextOffset(
    RvSdpListIter*    i,
    RvUint32*         time,
    RvSdpTimeUnit*    t_unit);
```

PARAMETERS

iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to thervSdpRepeatIntervalGetFirstOffset() function.

time

The offset time interval.

t_unit

The offset time units.

RETURN VALUES

Returns RV_TRUE if the next offset exists, or RV_FALSE otherwise.

rvSdpRepeatIntervalGetOffsetTime()

DESCRIPTION

Gets the session start offset time by index.

SYNTAX

```
RvUint32 rvSdpRepeatIntervalGetOffsetTime(
    RvSdpRepeatInterval*    repeat,
    RvSize_t                i);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

i

The index of the offset.

RETURN VALUES

Returns the session start offset time by index.

rvSdpRepeatIntervalGetOffsetUnits()

DESCRIPTION

Gets the session start offset time units by index.

SYNTAX

```
RvSdpTimeUnit rvSdpRepeatIntervalGetOffsetUnits(
    RvSdpRepeatInterval*    repeat,
    RvSize_t                i);
```

PARAMETERS

repeat

A pointer to the *RvSdpRepeatInterval*.

i

The index of the offset.

RETURN VALUES

Returns the session start offset time units by index.

rvSdpRepeatIntervalGetDurationUnits()

DESCRIPTION

Gets the active duration units of the session.

SYNTAX

```
RvSdpTimeUnit rvSdpRepeatIntervalGetDurationUnits(
    const RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the active duration units of the session.

rvSdpRepeatIntervalSetDurationUnits()

DESCRIPTION

Sets the active duration units of the session.

SYNTAX

```
void rvSdpRepeatIntervalSetDurationUnits(
    RvSdpRepeatInterval*    interv,
    RvSdpTimeUnit           unit);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

unit

The active duration units of the session.

RETURN VALUES

None.

rvSdpRepeatIntervalGetDurationTime()

DESCRIPTION

Gets the active duration time length of the session.

SYNTAX

```
RvUint32 rvSdpRepeatIntervalGetDurationTime(
    const RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the active duration time length of the session.

rvSdpRepeatIntervalSetDurationTime()

DESCRIPTION

Sets the active duration time length of the session.

SYNTAX

```
void rvSdpRepeatIntervalSetDurationTime(
    RvSdpRepeatInterval*    interv,
    RvUint32                time);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

time

The active duration time length of the session.

RETURN VALUES

None.

rvSdpRepeatIntervalGetIntervalUnits()

DESCRIPTION

Gets an *RvSdpRepeatInterval* time units.

SYNTAX

```
RvSdpTimeUnit rvSdpRepeatIntervalGetIntervalUnits(
    const RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the *RvSdpRepeatInterval* time units of the session.

rvSdpRepeatIntervalSetIntervalUnits()

DESCRIPTION

Sets an *RvSdpRepeatInterval* length units of the session.

SYNTAX

```
void rvSdpRepeatIntervalSetIntervalUnits(
    RvSdpRepeatInterval*    interv,
    RvSdpTimeUnit           unit);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

unit

The *RvSdpRepeatInterval* length units of the session.

RETURN VALUES

None.

rvSdpRepeatIntervalGetIntervalTime()

DESCRIPTION

Gets an *RvSdpRepeatInterval* time length of the session.

SYNTAX

```
RvUint32 rvSdpRepeatIntervalGetIntervalTime(
    const RvSdpRepeatInterval*    interv);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

RETURN VALUES

Returns the time length of the session.

rvSdpRepeatIntervalSetIntervalTime()

DESCRIPTION

Sets an *RvSdpRepeatInterval* time length of the session.

SYNTAX

```
void rvSdpRepeatIntervalSetIntervalTime(
    RvSdpRepeatInterval*    interv,
    RvUint32                time);
```

PARAMETERS

interv

A pointer to the *RvSdpRepeatInterval*.

time

The *RvSdpRepeatInterval* time length of the session.

RETURN VALUES

None.

14

TIME ZONE ADJUST FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpTimeZoneAdjust* objects. The *RvSdpTimeZoneAdjust* Type represents the time zone adjustments ('z=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpTimeZoneAdjustConstruct() **Obsolete**
- rvSdpTimeZoneAdjustConstructA() **Obsolete**
- rvSdpTimeZoneAdjustConstructCopy() **Obsolete**
- rvSdpTimeZoneAdjustConstructCopyA() **Obsolete**
- rvSdpTimeZoneAdjustDestruct()
- rvSdpTimeZoneAdjustCopy()

rvSdpTimeZoneAdjustConstruct()

DESCRIPTION

Constructs an *RvSdpTimeZoneAdjust*.

This function is obsolete. Please use rvSdpMsgTimeAddZoneAdjustment() instead.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpTimeZoneAdjustConstruct(
    RvSdpTimeZoneAdjust*    timeZone,
    RvUint32                t,
    RvInt32                 offsetTime,
    RvSdpTimeUnit           offsetUnits);
```

PARAMETERS

timeZone

A pointer to a valid *RvSdpTimeZoneAdjust*.

t

The time of the time shift.

offsetTime

The offset time.

offsetUnits

The units of the offset.

RETURN VALUES

Returns a pointer to the constructed *RvSdpTimeZoneAdjust*, or NULL if the function fails.

rvSdpTimeZoneAdjustConstructA()

DESCRIPTION

Constructs an *RvSdpTimeZoneAdjust*.

This function is obsolete. Please use rvSdpMsgTimeAddZoneAdjustment() instead.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpTimeZoneAdjustConstructA(
    RvSdpTimeZoneAdjust*    timeZone,
    RvUint32                t,
    RvInt32                 offsetTime,
    RvSdpTimeUnit           offsetUnits,
    RvAlloc*                a);
```

PARAMETERS

timeZone

A pointer to a valid *RvSdpTimeZoneAdjust*.

t

The time of the time shift.

offsetTime

The offset time.

offsetUnits

The units of the offset.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpTimeZoneAdjust*, or NULL if the function fails.

rvSdpTimeZoneAdjustConstructCopy()

DESCRIPTION

Constructs an *RvSdpTimeZoneAdjust* and copies the values from a source RvSdpTimeZoneAdjust.

This function is obsolete. Please use rvSdpMsgTimeAddZoneAdjustment() instead.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpTimeZoneAdjustConstructCopy(
    RvSdpTimeZoneAdjust*          dest,
    const RvSdpTimeZoneAdjust*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpTimeZoneAdjust* to be constructed. This parameter must point to valid memory.

src

The source*RvSdpTimeZoneAdjust*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpTimeZoneAdjust*, or NULL if the function fails.

rvSdpTimeZoneAdjustConstructCopyA()

DESCRIPTION

Constructs an t*RvSdpTimeZoneAdjust* and copies the values from a source*RvSdpTimeZoneAdjust*.

This function is obsolete. Please use rvSdpMsgTimeAddZoneAdjustment() instead.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpTimeZoneAdjustConstructCopyA(
    RvSdpTimeZoneAdjust*          dest,
    const RvSdpTimeZoneAdjust*    src,
    RvAlloc*                      a);
```

PARAMETERS

dest

A pointer to the *RvSdpTimeZoneAdjust* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpTimeZoneAdjust*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpTimeZoneAdjust*, or NULL if the function fails.

rvSdpTimeZoneAdjustDestruct()

DESCRIPTION

Destructs an *RvSdpTimeZoneAdjust*.

SYNTAX

```
void rvSdpTimeZoneAdjustDestruct(
    RvSdpTimeZoneAdjust*    timeZone);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

RETURN VALUES

None.

rvSdpTimeZoneAdjustCopy()

DESCRIPTION

Copies the values from a source *RvSdpTimeZoneAdjust* to a destination *RvSdpTimeZoneAdjust*.

SYNTAX

```
RvSdpTimeZoneAdjust* rvSdpTimeZoneAdjustCopy(
    RvSdpTimeZoneAdjust*          dest,
    const RvSdpTimeZoneAdjust*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpTimeZoneAdjust*. This parameter must point to constructed *RvSdpTimeZoneAdjust.*

src

The *RvSdpTimeZoneAdjust*.

RETURN VALUES

Returns a pointer to the destination *RvSdpTimeZoneAdjust*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpTimeZoneAdjustGetTime()
- rvSdpTimeZoneAdjustSetTime()
- rvSdpTimeZoneAdjustGetOffsetTime()
- rvSdpTimeZoneAdjustSetOffsetTime()
- rvSdpTimeZoneAdjustGetOffsetUnits()
- rvSdpTimeZoneAdjustSetOffsetUnits()

rvSdpTimeZoneAdjustGetTime()

DESCRIPTION

Gets the time of the time-shift event.

SYNTAX

```
RvUint32 rvSdpTimeZoneAdjustGetTime(
    const RvSdpTimeZoneAdjust*    timeZone);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

RETURN VALUES

Returns the time of the time-shift event.

rvSdpTimeZoneAdjustSetTime()

DESCRIPTION

Sets the time of the time-shift event.

SYNTAX

```
void rvSdpTimeZoneAdjustSetTime(
    RvSdpTimeZoneAdjust*    timeZone,
    RvUint32                t);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

t

The time of the time-shift event.

RETURN VALUES

None.

rvSdpTimeZoneAdjustGetOffsetTime()

DESCRIPTION

Gets the length of the time-shift.

SYNTAX

```
RvInt32 rvSdpTimeZoneAdjustGetOffsetTime(
    const RvSdpTimeZoneAdjust*    timeZone);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

RETURN VALUES

Returns the length of the time-shift.

rvSdpTimeZoneAdjustSetOffsetTime()

DESCRIPTION

Sets the length of the time-shift.

SYNTAX

```
void rvSdpTimeZoneAdjustSetOffsetTime(
    RvSdpTimeZoneAdjust*    timeZone,
    RvInt32                 offsetTime);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

offsetTime

The new offset length.

RETURN VALUES

None.

rvSdpTimeZoneAdjustGetOffsetUnits()

DESCRIPTION

Gets the units of the time-shift length.

SYNTAX

```
RvSdpTimeUnit rvSdpTimeZoneAdjustGetOffsetUnits(
    const RvSdpTimeZoneAdjust*    timeZone);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

RETURN VALUES

Returns the units of the time-shift length.

rvSdpTimeZoneAdjustSetOffsetUnits()

DESCRIPTION

Sets the units of the time-shift length.

SYNTAX

```
void rvSdpTimeZoneAdjustSetOffsetUnits(
    RvSdpTimeZoneAdjust*    timeZone,
    RvSdpTimeUnit           offsetUnits);
```

PARAMETERS

timeZone

A pointer to the *RvSdpTimeZoneAdjust*.

offsetUnits

The new offset length units.

RETURN VALUES

None.

15

KEY FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpKey* objects. The RvSdpKey Type represents the key ('k=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpKeyConstruct() **Obsolete**
- rvSdpKeyConstructA() **Obsolete**
- rvSdpKeyConstructCopy() **Obsolete**
- rvSdpKeyConstructCopyA() **Obsolete**
- rvSdpBadSyntaxKeyConstruct() amos **Obsolete**
- rvSdpBadSyntaxKeyConstructA() **Obsolete**
- rvSdpKeyIsBadSyntax()
- rvSdpKeyDestruct()
- rvSdpKeyCopy()

rvSdpKeyConstruct()

DESCRIPTION

Constructs an *RvSdpKey*.

This function is obsolete. Please use rvSdpMsgSetKey() or rvSdpMediaDescrSetKey() instead.

SYNTAX

```
RvSdpKey* rvSdpKeyConstruct(
    RvSdpKey*          key,
    RvSdpEncrMethod    type,
    const char*        data);
```

PARAMETERS

key

A pointer to the valid *RvSdpKey*.

type

The encryption method type.

data

The encryption data.

RETURN VALUES

Returns a pointer to the constructed RvSdpKey, or NULL if the function fails.

rvSdpKeyConstructA()

DESCRIPTION

Constructs an *RvSdpKey*.

This function is obsolete. Please use rvSdpMsgSetKey() or rvSdpMediaDescrSetKey() instead.

SYNTAX

```
RvSdpKey* rvSdpKeyConstructA(
    RvSdpKey*          key,
    RvSdpEncrMethod    type,
    const char*        data,
    RvAlloc*           a);
```

PARAMETERS

key

A pointer to valid *RvSdpKey*.

type

The encryption method type.

data

The encryption data.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpKey*, or NULL if the function fails.

rvSdpKeyConstructCopy()

DESCRIPTION

Constructs an *RvSdpKey* and copies the values from a source *RvSdpKey* field.

This function is obsolete. Please use rvSdpMsgSetKey() or rvSdpMediaDescrSetKey() instead.

SYNTAX

```
RvSdpKey* rvSdpKeyConstructCopy(
    RvSdpKey*          dest,
    const RvSdpKey*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpKey* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpKey*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpKey*, or NULL if the function fails.

rvSdpKeyConstructCopyA()

DESCRIPTION

Constructs an *RvSdpKey* and copies the values from a source *RvSdpKey* field.

This function is obsolete. Please use rvSdpMsgSetKey() or rvSdpMediaDescrSetKey() instead.

SYNTAX

```
RvSdpKey* rvSdpKeyConstructCopyA(
    RvSdpKey*          dest,
    const RvSdpKey*    src,
    RvAlloc*           a);
```

PARAMETERS

dest

A pointer to the *RvSdpKey* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpKey*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpKey*, or NULL if the function fails.

rvSdpBadSyntaxKeyConstruct() amos

DESCRIPTION

Constructs an *RvSdpKey* with proprietary format.

This function is obsolete. Please use rvSdpMsgSetBadSyntaxKey() or rvSdpMediaDescrSetBadSyntaxKey() instead.

SYNTAX

```
RvSdpKey* rvSdpBadSyntaxKeyConstruct(
    RvSdpKey*       key,
    const char*     badSyn);
```

PARAMETERS

key

A pointer to the valid *RvSdpKey*.

badSyn

The proprietary format of the repeat interval.

RETURN VALUES

Returns a pointer to the constructed *RvSdpKey*, or NULL if the function fails.

rvSdpBadSyntaxKeyConstructA()

DESCRIPTION

Constructs an *RvSdpKey* with proprietary format. **This function is obsolete.** Please use rvSdpMsgSetBadSyntaxKey() or rvSdpMediaDescrSetBadSyntaxKey() instead.

SYNTAX

```
RvSdpKey* rvSdpBadSyntaxKeyConstructA(
    RvSdpKey*       key,
    const char*     badSyn,
    RvAlloc*        a);
```

PARAMETERS

key

A pointer to the valid *RvSdpKey*.

badSyn

The proprietary format of the *RvSdpKey*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpKey*, or NULL if the function fails.

rvSdpKeyIsBadSyntax()

DESCRIPTION

Indicates whether or not the encryption *RvSdpKey* field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpKeyIsBadSyntax(
    RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpKeyDestruct()

DESCRIPTION

Destructs an *RvSdpKey*.

SYNTAX

```
void rvSdpKeyDestruct(
    RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

None.

rvSdpKeyCopy()

DESCRIPTION

Copies the values from a source *RvSdpKey* to a destination *RvSdpKey*.

SYNTAX

```
RvSdpKey* rvSdpKeyCopy(
    RvSdpKey*          dest,
    const RvSdpKey*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpKey*. This parameter must point to a constructed *RvSdpKey*.

src

The *RvSdpKey*.

RETURN VALUES

Returns a pointer to the destination *RvSdpKey*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpKeyGetBadSyntax()
- rvSdpKeySetBadSyntax()
- rvSdpKeyGetType()
- rvSdpKeySetType()
- rvSdpKeyGetTypeStr()
- rvSdpKeySetTypeStr()
- rvSdpKeyGetData()
- rvSdpKeySetData()

rvSdpKeyGetBadSyntax()

DESCRIPTION

Gets a proprietary formatted encryption *RvSdpKey* field value, or empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpKeyGetBadSyntax(
    const RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

Returns the bad syntax value.

rvSdpKeySetBadSyntax()

DESCRIPTION

Sets the SDP encryption *RvSdpKey* field value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpKeySetBadSyntax(
    RvSdpKey*       o,
    const char*     bs);
```

PARAMETERS

o

A pointer to the *RvSdpKey.*

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpKeyGetType()

DESCRIPTION

Gets the encryption type of an *RvSdpKey*.

SYNTAX

```
RvSdpEncrMethod rvSdpKeyGetType(
    const RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

Returns the *RvSdpKey* encryption type.

rvSdpKeySetType()

DESCRIPTION

Sets the encryption type of an *RvSdpKey*

SYNTAX

```
void rvSdpKeySetType(
    RvSdpKey*          key,
    RvSdpEncrMethod    type);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

type

The encryption type.

RETURN VALUES

None.

rvSdpKeyGetTypeStr()

DESCRIPTION

Gets the encryption type string of an *RvSdpKey*.

SYNTAX

```
const char* rvSdpKeyGetTypeStr(
    RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

Returns the *RvSdpKey* encryption type string.

rvSdpKeySetTypeStr()

DESCRIPTION

Sets the encryption type string of an *RvSdpKey*.

SYNTAX

```
RvSdpStatus rvSdpKeySetTypeStr(
    RvSdpKey*       key,
    const char*     typeStr);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

typeStr

The encryption type string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpKeyGetData()

DESCRIPTION

Gets the encryption data of an *RvSdpKey*.

SYNTAX

```
const char* rvSdpKeyGetData(
    const RvSdpKey*    key);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

RETURN VALUES

Returns the *RvSdpKey* encryption data.

rvSdpKeySetData()

DESCRIPTION

Sets the encryption data of an *RvSdpKey*.

SYNTAX

```
RvSdpStatus rvSdpKeySetData(
    RvSdpKey*       key,
    const char*     data);
```

PARAMETERS

key

A pointer to the *RvSdpKey*.

data

The encryption data.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

16

ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpAttribute* objects. The *RvSdpAttribute* Type represents the attribute ('a=') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpAttributeConstruct() **Obsolete**
- rvSdpAttributeConstructA() **Obsolete**
- rvSdpAttributeConstructCopy() **Obsolete**
- rvSdpAttributeConstructCopyA() **Obsolete**
- rvSdpAttributeDestruct()
- rvSdpAttributeCopy()

rvSdpAttributeConstruct()

DESCRIPTION

Constructs an *RvSdpAttribute* using the default *RvAlloc*.

This function is obsolete. Please use rvSdpMsgAddAttr() or rvSdpMediaDescrAddAttr() instead.

SYNTAX

```
RvSdpAttribute* rvSdpAttributeConstruct(
    RvSdpAttribute*    attr,
    const char*        name,
    const char*        value);
```

PARAMETERS

attr

A pointer to a valid *RvSdpAttribute*.

name

The name of the *RvSdpAttribute*.

value

The value of the *RvSdpAttribute*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpAttribute*, or NULL if the function fails.

rvSdpAttributeConstructA()

DESCRIPTION

Constructs an *RvSdpAttribute*.

This function is obsolete. Please use rvSdpMsgAddAttr() or rvSdpMediaDescrAddAttr() instead.

SYNTAX

```
RvSdpAttribute* rvSdpAttributeConstructA(
    RvSdpAttribute*    attr,
    const char*        name,
    const char*        value,
    RvAlloc*           a);
```

PARAMETERS

attr

A pointer to a valid *RvSdpAttribute*.

name

The name of the *RvSdpAttribute*.

value

The value of the *RvSdpAttribute*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpAttribute*, or NULL if the function fails.

rvSdpAttributeConstructCopy()

DESCRIPTION

Constructs an *RvSdpAttribute* and copies the values from a source *RvSdpAttribute*. **This function is obsolete.** Please use rvSdpMsgAddAttr() or rvSdpMediaDescrAddAttr() instead.

SYNTAX

```
RvSdpAttribute* rvSdpAttributeConstructCopy(
    RvSdpAttribute*          dest,
    const RvSdpAttribute*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpAttribute* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpAttribute*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpAttribute*, or NULL if the function fails.

rvSdpAttributeConstructCopyA()

DESCRIPTION

Constructs an *RvSdpAttribute* and copies the values from a source *RvSdpAttribute*.

This function is obsolete. Please use rvSdpMsgAddAttr() or rvSdpMediaDescrAddAttr() instead.

SYNTAX

```
RvSdpAttribute* rvSdpAttributeConstructCopyA(
    RvSdpAttribute*          dest,
    const RvSdpAttribute*    src,
    RvAlloc*                 a);
```

PARAMETERS

dest

A pointer to *RvSdpAttribute* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpAttribute*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpAttribute*, or NULL if the function fails.

rvSdpAttributeDestruct()

DESCRIPTION

Destructs an *RvSdpAttribute*.

SYNTAX

```
void rvSdpAttributeDestruct(
    RvSdpAttribute*    attr);
```

PARAMETERS

attr

A pointer to the *RvSdpAttribute*.

RETURN VALUES

None.

rvSdpAttributeCopy()

DESCRIPTION

Copies the values from a source *RvSdpAttribute* to a destination *RvSdpAttribute*. Only generic attributes can be copied in this way. If one of the arguments is a special *RvSdpAttribute*, NULL will be returned.

SYNTAX

```
RvSdpAttribute* rvSdpAttributeCopy(
    RvSdpAttribute*          dest,
    const RvSdpAttribute*    src);
```

PARAMETERS

dest

A pointer to the destination generic *RvSdpAttribute*. This parameter must point to a constructed *RvSdpAttribute*.

src

The source generic *RvSdpAttribute*.

RETURN VALUES

Returns a pointer to the destination *RvSdpAttribute*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpAttributeGetName()
- rvSdpAttributeSetName()
- rvSdpAttributeGetValue()
- rvSdpAttributeSetValue()

rvSdpAttributeGetName()

DESCRIPTION

Gets the name of an *RvSdpAttribute*.

SYNTAX

```
const char* rvSdpAttributeGetName(
    RvSdpAttribute*    attr);
```

PARAMETERS

attr

A pointer to the *RvSdpAttribute*.

RETURN VALUES

Returns the *RvSdpAttribute* name.

rvSdpAttributeSetName()

DESCRIPTION

Sets the name of an *RvSdpAttribute*.

SYNTAX

```
RvSdpStatus rvSdpAttributeSetName(
    RvSdpAttribute*    a,
    const char*        name);
```

PARAMETERS

attr

A pointer to the *RvSdpAttribute*.

name

The name of the new *RvSdpAttribute*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpAttributeGetValue()

DESCRIPTION

Gets the value of an *RvSdpAttribute*, or an empty string ("") if the value is not set.

SYNTAX

```
const char* rvSdpAttributeGetValue(
    RvSdpAttribute*    attr);
```

PARAMETERS

attr

A pointer to the *RvSdpAttribute*.

RETURN VALUES

Returns the *RvSdpAttribute* value, or the empty string if the value is not set.

rvSdpAttributeSetValue()

DESCRIPTION

Sets the value of an *RvSdpAttribute*.

SYNTAX

```
vSdpStatus rvSdpAttributeSetValue(
    RvSdpAttribute*    attr,
    const char*        value);
```

PARAMETERS

attr

A pointer to the *RvSdpAttribute*.

name

The value of the new *RvSdpAttribute*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

17

RTP MAP ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpRtpMap* objects. The *RvSdpRtpMap* Type represents the RTP map attribute ('a=rtpmap:') field of an *RvSdpMsg*.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpRtpMapConstruct() **Obsolete**
- rvSdpRtpMapConstructA() **Obsolete**
- rvSdpRtpMapConstructCopy() **Obsolete**
- rvSdpRtpMapConstructCopyA() **Obsolete**
- rvSdpBadSyntaxRtpMapConstruct() **Obsolete**
- rvSdpBadSyntaxRtpMapConstructA() **Obsolete**
- rvSdpRtpMapIsBadSyntax()
- rvSdpRtpMapDestruct()
- rvSdpRtpMapCopy()

rvSdpRtpMapConstruct()

DESCRIPTION

Constructs an *RvSdpRtpMap*.

This function is obsolete. Please use rvSdpMsgAddRtpMap() or rvSdpMediaDescrAddRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapConstruct(
    RvSdpRtpMap*    rtpMap,
    int             payload,
    const char*     encoding_name,
    int             rate);
```

PARAMETERS

rtpMap

A pointer to a valid *RvSdpRtpMap*.

payload

The payload number of the *RvSdpRtpMap*.

encoding_name

The encoding name of the *RvSdpRtpMap*.

rate

The rate value of the *RvSdpRtpMap*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpRtpMapConstructA()

DESCRIPTION

Constructs an *RvSdpRtpMap*.

This function is obsolete. Please use rvSdpMsgAddRtpMap() or rvSdpMediaDescrAddRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapConstructA(
    RvSdpRtpMap*    rtpMap,
    int             payload,
    const char*     encoding_name,
    int             rate,
    RvAlloc*        alloc);
```

PARAMETERS

rtpMap

A pointer to a valid *RvSdpRtpMap*.

payload

The payload number of the *RvSdpRtpMap*.

encoding_name

The encoding name of the *RvSdpRtpMap*.

rate

The rate value of the *RvSdpRtpMap*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpRtpMapConstructCopy()

DESCRIPTION

Constructs an *RvSdpRtpMap* and copies the values from a source *RvSdpRtpMap* field.

This function is obsolete. Please use rvSdpMsgAddRtpMap() or rvSdpMediaDescrAddRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapConstructCopy(
    RvSdpRtpMap*          dest,
    const RvSdpRtpMap*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpRtpMap* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpRtpMap*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpRtpMapConstructCopyA()

DESCRIPTION

Constructs an *RvSdpRtpMap* and copies the values from a source *RvSdpRtpMap* field.

This function is obsolete. Please use rvSdpMsgAddRtpMap() or rvSdpMediaDescrAddRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapConstructCopyA(
    RvSdpRtpMap*          dest,
    const RvSdpRtpMap*    src,
    RvAlloc*              alloc);
```

PARAMETERS

dest

A pointer to the *RvSdpRtpMap* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpRtpMap*.

alloc

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpBadSyntaxRtpMapConstruct()

DESCRIPTION

Constructs an *RvSdpRtpMap* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxRtpMap() or rvSdpMediaDescrAddBadSyntaxRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpBadSyntaxRtpMapConstruct(
    RvSdpRtpMap*    rtpMap,
    const char*     badSyn);
```

PARAMETERS

rtpMap

A pointer to a valid *RvSdpRtpMap*.

badSyn

The proprietary format of the *RvSdpRtpMap* field.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpBadSyntaxRtpMapConstructA()

DESCRIPTION

Constructs an *RvSdpRtpMap* with proprietary format.

This function is obsolete. Please use rvSdpMsgAddBadSyntaxRtpMap() or rvSdpMediaDescrAddBadSyntaxRtpMap() instead.

SYNTAX

```
RvSdpRtpMap* rvSdpBadSyntaxRtpMapConstructA(
    RvSdpRtpMap*    rtpMap,
    const char*     badSyn,
    RvAlloc*        alloc);
```

PARAMETERS

rtpMap

A pointer to a valid *RvSdpRtpMap*.

badSyn

The proprietary format of the *RvSdpRtpMap* field.

alloc

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpRtpMap*, or NULL if the function fails.

rvSdpRtpMapIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpRtpMap* attribute field is proprietary-formatted.

SYNTAX

```
RvBool rvSdpRtpMapIsBadSyntax(
    RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

rvSdpRtpMapDestruct()

DESCRIPTION

Destructs an *RvSdpRtpMap*.

SYNTAX

```
void rvSdpRtpMapDestruct(
    RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

None.

rvSdpRtpMapCopy()

DESCRIPTION

Copies the values from a source *RvSdpRtpMap* to destination *RvSdpRtpMap*.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapCopy(
    RvSdpRtpMap*          dest,
    const RvSdpRtpMap*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpRtpMap*. This parameter must point to a constructed *RvSdpRtpMap*.

src

The *RvSdpRtpMap*.

RETURN VALUES

Returns a pointer to the destination *RvSdpRtpMap*, or NULL if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpRtpMapGetBadSyntax()
- rvSdpRtpMapSetBadSyntax()
- rvSdpRtpMapGetChannels()
- rvSdpRtpMapSetChannels()
- rvSdpRtpMapGetPayload()
- rvSdpRtpMapSetPayload()
- rvSdpRtpMapGetEncodingName()
- rvSdpRtpMapSetEncodingName()
- rvSdpRtpMapGetClockRate()
- rvSdpRtpMapSetClockRate()
- rvSdpRtpMapGetEncodingParameters()
- rvSdpRtpMapSetEncodingParameters()

rvSdpRtpMapGetBadSyntax()

DESCRIPTION

Gets a proprietary format of an *RvSdpRtpMap* attribute value, or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpRtpMapGetBadSyntax(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the bad syntax value.

rvSdpRtpMapSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpRtpMap* attribute value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpRtpMapSetBadSyntax(
    RvSdpRtpMap*    o,
    const char*     bs);
```

PARAMETERS

o

A pointer to the *RvSdpRtpMap*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpRtpMapGetChannels()

DESCRIPTION

Gets the number of channels. This function is used for media type audio instead of rvSdpRtpMapGetEncodingParameters().

SYNTAX

```
int rvSdpRtpMapGetChannels(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the number of channels.

rvSdpRtpMapSetChannels()

DESCRIPTION

Sets the number of channels. This function is used for media type audio instead of rvSdpRtpMapSetEncodingParameters().

SYNTAX

```
RvSdpStatus rvSdpRtpMapSetChannels(
    RvSdpRtpMap*    rtpMap,
    int             channels);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

channels

The number of channels.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpRtpMapGetPayload()

DESCRIPTION

Gets the payload number.

SYNTAX

```
int rvSdpRtpMapGetPayload(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the payload number.

rvSdpRtpMapSetPayload()

DESCRIPTION

Sets the payload number.

SYNTAX

```
void rvSdpRtpMapSetPayload(
    RvSdpRtpMap*    rtpMap,
    int             payload);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

payload

The payload number.

RETURN VALUES

None.

rvSdpRtpMapGetEncodingName()

DESCRIPTION

Gets the encoding name of an *RvSdpRtpMap*.

SYNTAX

```
const char* rvSdpRtpMapGetEncodingName(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the *RvSdpRtpMap* encoding name.

rvSdpRtpMapSetEncodingName()

DESCRIPTION

Sets the encoding name of an *RvSdpRtpMap*.

SYNTAX

```
RvSdpStatus rvSdpRtpMapSetEncodingName(
    RvSdpRtpMap*    rtpMap,
    const char*     name);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

name

The encoding name of the *RvSdpRtpMap*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpRtpMapGetClockRate()

DESCRIPTION

Gets the clock-rate of an *RvSdpRtpMap*.

SYNTAX

```
RvUint32 rvSdpRtpMapGetClockRate(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the *RvSdpRtpMap* clock-rate.

rvSdpRtpMapSetClockRate()

DESCRIPTION

Sets the clock-rate of an *RvSdpRtpMap*.

SYNTAX

```
void rvSdpRtpMapSetClockRate(
    RvSdpRtpMap*    rtpMap,
    RvUint32        rate);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

rate

The clock-rate of the *RvSdpRtpMap*.

RETURN VALUES

None.

rvSdpRtpMapGetEncodingParameters()

DESCRIPTION

Gets the encoding parameters of an *RvSdpRtpMap*.

SYNTAX

```
const char* rvSdpRtpMapGetEncodingParameters(
    const RvSdpRtpMap*    rtpMap);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

RETURN VALUES

Returns the *RvSdpRtpMap* encoding parameters.

rvSdpRtpMapSetEncodingParameters()

DESCRIPTION

Sets the encoding parameters of an *RvSdpRtpMap*.

SYNTAX

RvSdpStatus rvSdpRtpMapSetEncoding`Parameters(`

```
    RvSdpRtpMap*     rtpMap,
    const char*      s);
```

PARAMETERS

rtpMap

A pointer to the *RvSdpRtpMap*.

s

The encoding parameters of the *RvSdpRtpMap*.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

18

KEY MANAGEMENT ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

The key management attribute is used to transfer the key management protocol and the base64 encoded key value. RV_SDPKEYMGMT_MIKEY ("mikey") is the only currently registered value of key management protocol.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpKeyMgmtDecodeKeyData()
- rvSdpKeyMgmtIsBadSyntax()

rvSdpKeyMgmtDecodeKeyData()

DESCRIPTION

Decodes key data (using B64 decoding) of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
RvSize_t rvSdpKeyMgmtDecodeKeyData(
    RvSdpKeyMgmtAttr*    keyMgmt,
    unsigned char*       decodedData,
    int                  dataLen);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

decodedData

The output buffer for B64 decoding.

dataLen

The size of the *decodedData* buffer.

RETURN VALUES

Returns the size of the decoded buffer.

rvSdpKeyMgmtIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpKeyMgmtAttr* is proprietary-formatted.

SYNTAX

```
RvBool rvSdpKeyMgmtIsBadSyntax(
    RvSdpKeyMgmtAttr*    keyMgmt);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpKeyMgmtGetPrtclIdTxt()
- rvSdpKeyMgmtSetPrtclIdTxt()
- rvSdpKeyMgmtGetPrtclId()
- rvSdpKeyMgmtSetPrtclId()
- rvSdpKeyMgmtGetKeyData()
- rvSdpKeyMgmtSetKeyData()
- rvSdpKeyMgmtGetBadSyntax()
- rvSdpKeyMgmtSetBadSyntax()

rvSdpKeyMgmtGetPrtclIdTxt()

DESCRIPTION

Gets the protocol ID text value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
const char* rvSdpKeyMgmtGetPrtclIdTxt(
    const RvSdpKeyMgmtAttr*    keyMgmt);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpKeyMgmtAttr*.

rvSdpKeyMgmtSetPrtclIdTxt()

DESCRIPTION

Sets the protocol ID text value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
RvSdpStatus rvSdpKeyMgmtSetPrtclIdTxt(
    RvSdpKeyMgmtAttr*    keyMgmt,
    const char*          prtclId);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

prtclId

The new value of the protocol ID text.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpKeyMgmtGetPrtclId()

DESCRIPTION

Gets the protocol ID value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
RvSdpKeyMgmtPrtclType rvSdpKeyMgmtGetPrtclId(
    const RvSdpKeyMgmtAttr*    keyMgmt);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpKeyMgmtAttr*.

rvSdpKeyMgmtSetPrtclId()

DESCRIPTION

Sets the protocol ID value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
RvSdpStatus rvSdpKeyMgmtSetPrtclId(
    RvSdpKeyMgmtAttr*        keyMgmt,
    RvSdpKeyMgmtPrtclType    prtclId);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

prtclId

The new value of the protocol ID.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpKeyMgmtGetKeyData()

DESCRIPTION

Gets the encryption key data value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
const char* rvSdpKeyMgmtGetKeyData(
    const RvSdpKeyMgmtAttr*    keyMgmt);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpKeyMgmtAttr*.

rvSdpKeyMgmtSetKeyData()

DESCRIPTION

Sets the key data value of an *RvSdpKeyMgmtAttr*.

SYNTAX

```
RvSdpStatus rvSdpKeyMgmtSetKeyData(
    RvSdpKeyMgmtAttr*    keyMgmt,
    const char*          keyData);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

keyData

The new value of the protocol ID.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpKeyMgmtGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpKeyMgmtAttr* value or an empty string ("") if the value is legal.

SYNTAX

```
const char* rvSdpKeyMgmtGetBadSyntax(
    const RvSdpKeyMgmtAttr*    keyMgmt);
```

PARAMETERS

keyMgmt

A pointer to the *RvSdpKeyMgmtAttr*.

RETURN VALUES

Returns the bad syntax value.

rvSdpKeyMgmtSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpKeyMgmtAttr* value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpKeyMgmtSetBadSyntax(
    const RvSdpKeyMgmtAttr*    o,
    const char*                bs);
```

PARAMETERS

o

A pointer to the *RvSdpKeyMgmtAttr*.

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

19

CRYPTO ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpCryptoAttr* objects. The *RvSdpCryptoAttr* Type represents the crypto attribute ('a=crypto:') field of an *RvSdpMsg*. These functions can be used only if the RV_SDP_CRYPTO_ATTR compilation switch (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpCryptoAddKeyParam()
- rvSdpCryptoRemoveKeyParam()
- rvSdpCryptoClearKeyParams()
- rvSdpCryptoAddSessionParam()
- rvSdpCryptoRemoveSessionParam()
- rvSdpCryptoClearSessionParams()
- rvSdpCryptoIsBadSyntax()

rvSdpCryptoAddKeyParam()

DESCRIPTION

Adds the other pair of key parameters of an *RvSdpCryptoAttr*.

SYNTAX

```
RvSdpStatus rvSdpCryptoAddKeyParam(
    RvSdpCryptoAttr*    crypto,
    const char*         method,
    const char*         info);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

method

The key method to be added.

info

The key information to be added.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpCryptoRemoveKeyParam()

DESCRIPTION

Removes (and destructs) a key parameter of an *RvSdpCryptoAttr* by index.

SYNTAX

```
void rvSdpCryptoRemoveKeyParam(
    RvSdpCryptoAttr*    crypto,
    RvSize_t            index);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpCryptoGetNumOfKeyParams().

RETURN VALUES

None.

rvSdpCryptoClearKeyParams()

DESCRIPTION

Removes (and destructs) all key parameters set in an *RvSdpCryptoAttr*.

SYNTAX

```
void rvSdpCryptoClearKeyParams(
    RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

None.

rvSdpCryptoAddSessionParam()

DESCRIPTION

Adds another session parameter of an *RvSdpCryptoAttr*.

SYNTAX

```
RvSdpStatus rvSdpCryptoAddSessionParam(
    RvSdpCryptoAttr*    crypto,
    const char*         spar);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

spar

The session parameter to be added.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpCryptoRemoveSessionParam()

DESCRIPTION

Removes (and destructs) an *RvSdpCryptoAttr* parameter by index.

SYNTAX

```
void rvSdpCryptoRemoveSessionParam(
    RvSdpCryptoAttr*    crypto,
    RvSize_t            index);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling calling rvSdpCryptoGetNumOfKeyParams().

RETURN VALUES

None.

rvSdpCryptoClearSessionParams()

DESCRIPTION

Removes (and destructs) all session parameters set in an *RvSdpCryptoAttr*.

SYNTAX

```
void rvSdpCryptoClearSessionParams(
    RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

None.

rvSdpCryptoIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpCryptoAttr* is proprietary-formatted.

SYNTAX

```
RvBool rvSdpCryptoIsBadSyntax(
    RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

RETURN VALUES

Returns RV_TRUE if the field is bad syntax, or RV_FALSE otherwise.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpCryptoGetTag()
- rvSdpCryptoSetTag()
- rvSdpCryptoGetSuite()
- rvSdpCryptoSetSuite()
- rvSdpCryptoGetNumOfKeyParams()
- rvSdpCryptoGetKeyMethod()
- rvSdpCryptoGetKeyInfo()
- rvSdpCryptoGetNumOfSessionParams()
- rvSdpCryptoGetSessionParam()
- rvSdpCryptoGetBadSyntax()
- rvSdpCryptoSetBadSyntax()

rvSdpCryptoGetTag()

DESCRIPTION

Gets the tag value of an *RvSdpCryptoAttr*.

SYNTAX

```
RvUint rvSdpCryptoGetTag(
    const RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpCryptoAttr*.

rvSdpCryptoSetTag()

DESCRIPTION

Sets the tag value of an *RvSdpCryptoAttr*.

SYNTAX

```
void rvSdpCryptoSetTag(
    RvSdpCryptoAttr*    crypto,
    RvUint              tag);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

tag

The new value of the *RvSdpCryptoAttr* tag.

RETURN VALUES

None.

rvSdpCryptoGetSuite()

DESCRIPTION

Gets the suite value of an *RvSdpCryptoAttr*.

SYNTAX

```
const char* rvSdpCryptoGetSuite(
    const RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpCryptoAttr*.

rvSdpCryptoSetSuite()

DESCRIPTION

Sets the suite value of an *RvSdpCryptoAttr*.

SYNTAX

```
RvSdpStatus rvSdpCryptoSetSuite(
    RvSdpCryptoAttr*    crypto,
    const char*         suite);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

suite

The new value of the suite.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpCryptoGetNumOfKeyParams()

DESCRIPTION

Gets the number of key parameters set in an *RvSdpCryptoAttr*.

SYNTAX

```
RvSize_t rvSdpCryptoGetNumOfKeyParams(
    const RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

Returns the number of key parameters set in the *RvSdpCryptoAttr*.

rvSdpCryptoGetKeyMethod()

DESCRIPTION

Gets a crypto method of an *RvSdpCryptoAttr* by index.

SYNTAX

```
const char* rvSdpCryptoGetKeyMethod(
    const RvSdpCryptoAttr*    crypto,
    RvSize_t                  index);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpCryptoGetNumOfKeyParams().

RETURN VALUES

Returns the requested *RvSdpCryptoAttr* method.

rvSdpCryptoGetKeyInfo()

DESCRIPTION

Gets the key information of an *RvSdpCryptoAttr* by index.

SYNTAX

```
const char* rvSdpCryptoGetKeyInfo(
    const RvSdpCryptoAttr*    crypto,
    RvSize_t                  index);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpCryptoGetNumOfKeyParams().

RETURN VALUES

Returns the requested key information.

rvSdpCryptoGetNumOfSessionParams()

DESCRIPTION

Gets the number of session parameters set in an *RvSdpCryptoAttr*.

SYNTAX

```
RvSize_t rvSdpCryptoGetNumOfSessionParams(
    const RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

RETURN VALUES

Returns the number of session parameters set in the *RvSdpCryptoAttr*.

rvSdpCryptoGetSessionParam()

DESCRIPTION

Gets a session parameter of an *RvSdpCryptoAttr* by index.

SYNTAX

```
const char* rvSdpCryptoGetSessionParam(
    const RvSdpCryptoAttr*    crypto,
    RvSize_t                  index);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr*.

index

The index, which should start at zero (0) and must be smaller than the number of elements in the list. The number of elements in the list is retrieved by calling rvSdpCryptoGetNumOfKeyParams().

RETURN VALUES

Returns the requested session parameter.

rvSdpCryptoGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpCryptoAttr* value or empty string if the value is legal.

SYNTAX

```
const char* rvSdpCryptoGetBadSyntax(
    const RvSdpCryptoAttr*    crypto);
```

PARAMETERS

crypto

A pointer to the *RvSdpCryptoAttr.*

RETURN VALUES

Returns the bad syntax value.

rvSdpCryptoSetBadSyntax()

DESCRIPTION

Sets the SDP *RvSdpCryptoAttr* value to proprietary format.

SYNTAX

```
RvSdpStatus rvSdpCryptoSetBadSyntax(
    RvSdpCryptoAttr*    o,
    const char*         bs);
```

PARAMETERS

o

A pointer to the *RvSdpCryptoAttr.*

bs

The proprietary-formatted value to be set.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

20

MEDIA GROUP ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpMediaGroupAttr* objects. The *RvSdpMediaGroupAttr* type represents the MediaGroup attribute ('a=group:') field of an *RvSdpMsg*. This attribute can appear only in the message context. These functions can be used only if the RV_SDP_MEDIA_GROUPING_ATTR compilation switch (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMediaGroupAttrAddMid()
- rvSdpMediaGroupAttrRemoveMid()
- rvSdpMediaGroupAttrClearMidParams()
- rvSdpMsgAddBadSyntaxMediaGroup()
- rvSdpMediaGroupAttrIsBadSyntax()
- rvSdpMediaDescrDestroyMidAttr()

rvSdpMediaGroupAttrAddMid()

DESCRIPTION

Adds one MidId parameter to the *RvSdpMediaGroupAttr* (iMidIdParams).

SYNTAX

```
RvSdpStatus rvSdpMediaGroupAttrAddMid(
    RvSdpMediaGroupAttr*    mediaGroupAttr,
    const char*             midTag);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

midTag

The MidId tag to be added.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if it fails.

rvSdpMediaGroupAttrRemoveMid()

DESCRIPTION

Removes one MidId parameter from the MidId parameters list by index.

SYNTAX

```
void rvSdpMediaGroupAttrRemoveMid(
    RvSdpMediaGroupAttr*    mediaGroupAttr,
    RvSize_t                index);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

index

An index to a specific cell in the MidId parameters list (iMidIdParams) that should be removed. The index should start at zero (0) and must be smaller than the number of elements in the MidId array. The number of elements in the array is retrieved by calling rvSdpMediaGroupAttrGetNumOfMidParams().

RETURN VALUES

None.

rvSdpMediaGroupAttrClearMidParams()

DESCRIPTION

Removes (and destructs) all MidId parameters set in the *RvSdpMediaGroupAttr*.

SYNTAX

```
void rvSdpMediaGroupAttrClearMidParams(
    RvSdpMediaGroupAttr*    mediaGroup);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

RETURN VALUES

None.

rvSdpMsgAddBadSyntaxMediaGroup()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpMediaGroupAttr* at the session level.

SYNTAX

```
RvSdpMediaGroupAttr* rvSdpMsgAddBadSyntaxMediaGroup(
    RvSdpMsg*       msg,
    const char*     badSyn);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

badSyn

The proprietary-formatted value to be set.

RETURN VALUES

Returns a pointer to the newly-created *RvSdpMediaGroupAttr* if the function succeeds, or a NULL pointer if it fails.

rvSdpMediaGroupAttrIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpMediaGroupAttr* is proprietary-formatted.

SYNTAX

```
RvBool rvSdpMediaGroupAttrIsBadSyntax(
    RvSdpMediaGroupAttr*    mediaGroupAttr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

RETURN VALUES

Returns RV_TRUE if the field contains bad syntax. Otherwise, returns RV_FALSE.

rvSdpMediaDescrDestroyMidAttr()

DESCRIPTION

Removes the Mid attribute from the *RvSdpMediaDescr*.

SYNTAX

```
void rvSdpMediaDescrDestroyMidAttr(
    RvSdpMediaDescr*    descr)
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr*.

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpMediaGroupAttrGetSemanticsStr()
- rvSdpMediaGroupAttrGetSemantics()
- rvSdpMediaGroupAttrSetSemanticsStr()
- rvSdpMediaGroupAttrSetSemantics()
- rvSdpMsgGetMediaDescrByGroup()
- rvSdpMediaGroupAttrGetMid()
- rvSdpMediaGroupAttrGetNumOfMidParams()
- rvSdpMsgGetNumOfMediaGroup()
- rvSdpMsgGetMediaGroup()
- rvSdpMediaGroupAttrGetBadSyntax()
- rvSdpMsgGetNumOfMediaGroup()
- rvSdpMsgGetFirstMediaGroup()
- rvSdpMsgGetNextMediaGroup()

rvSdpMediaGroupAttrGetSemanticsStr()

DESCRIPTION

Gets the semantics value of the *RvSdpMediaGroupAttr* as a string ("LS"/"FID"/ "SRF").

SYNTAX

```
const char* rvSdpMediaGroupAttrGetSemanticsStr(
    const RvSdpMediaGroupAttr*    mediaGroupAttr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr.*

RETURN VALUES

Returns the requested field of the *RvSdpMediaGroupAttr* as a string.

rvSdpMediaGroupAttrGetSemantics()

DESCRIPTION

Gets the semantics value of the *RvSdpMediaGroupAttr* as an enumeration ("LS"/"FID"/"SRF").

SYNTAX

```
const RvSdpGroupSemanticsType
rvSdpMediaGroupAttrGetSemantics(
    const RvSdpMediaGroupAttr*    mediaGroupAttr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpMediaGroupAttr* as an enumeration.

rvSdpMediaGroupAttrSetSemanticsStr()

DESCRIPTION

Sets the tag value of the *RvSdpMediaGroupAttr* as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaGroupAttrSetSemanticsStr(
    RvSdpMediaGroupAttr*    mediaGroupAttr,
    const char*             semanticsStr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

semanticsStr

The new semantics value of the *RvSdpMediaGroupAttr* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if it fails.

rvSdpMediaGroupAttrSetSemantics()

DESCRIPTION

Sets the tag value of *RvSdpMediaGroupAttr* as an enumeration.

SYNTAX

```
RvSdpStatus rvSdpMediaGroupAttrSetSemantics(
    RvSdpMediaGroupAttr*       mediaGroupAttr,
    RvSdpGroupSemanticsType    eSemantics)
```

PARAMETERS

mediaGroupAttr

A pointer to *RvSdpMediaGroupAttr*.

eSemantics

The new semantics value of *RvSdpMediaGroupAttr* as an enumeration.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMsgGetMediaDescrByGroup()

DESCRIPTION

Depending on the message and the *mediaGroupAttr* (a=group), this function returns a pointer to the mediaDesc labeled with a MidId tag that appears in the given index in the MidId parameter list of the Media Group attribute.

SYNTAX

```
RvSdpMediaDescr* rvSdpMsgGetMediaDescrByGroup(
    RvSdpMsg                *msg,
    RvSdpMediaGroupAttr*    mediaGroupAttr,
    RvUint                  index);
```

PARAMETERS

Msg

A pointer to an *RvSdpMsg* object.

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr.*

index

An index to a specific cell in the MidId param list (iMidIdParams).

RETURN VALUES

If successful, this function returns a pointer to the *RvSdpMediaDescr* that was found. NULL is returned if the mediaDescr with this MidId label was not found.

rvSdpMediaGroupAttrGetMid()

DESCRIPTION

Gets one MidId parameter of the Media Group special attribute by index.

SYNTAX

```
const char* rvSdpMediaGroupAttrGetMid(
    const RvSdpMediaGroupAttr*    mediaGroupAttr,
    RvSize_t                      index);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

index

An index to a specific cell in the MidId parameter list (iMidIdParams). The index should start at zero (0) and must be smaller than the number of elements in the MidId array. The number of elements in the array is retrieved by calling rvSdpMediaGroupAttrGetNumOfMidParams().

RETURN VALUES

Returns the requested Mid value information as a string.

rvSdpMediaGroupAttrGetNumOfMidParams()

DESCRIPTION

Gets the number of MidId parameters set in the *RvSdpMediaGroupAttr*.

SYNTAX

```
RvSize_t rvSdpMediaGroupAttrGetNumOfMidParams(
    const RvSdpMediaGroupAttr*    mediaGroupAttr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

RETURN VALUES

Returns the number MidId parameters set in the *RvSdpMediaGroupAttr* (returns iMidIdParamsNum).

rvSdpMsgGetNumOfMediaGroup()

DESCRIPTION

Gets the number of the *RvSdpMediaGroupAttr* set in the SDP message context.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfMediaGroup(
    const RvSdpMsg*    msg);
```

PARAMETERS

Msg

A pointer to an *RvSdpMsg* object.

RETURN VALUES

Returns the number of Media Group special attributes set in the *RvSdpMsg*.

rvSdpMsgGetMediaGroup()

DESCRIPTION

Gets an *RvSdpMediaGroupAttr* by index.

SYNTAX

```
RvSdpMediaGroupAttr* rvSdpMsgGetMediaGroup(
    const RvSdpMsg*    msg,
    RvSize_t           index);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg*.

index

The index, which should start at zero (0) and must be smaller than the number of Media Group elements in the list. The number of elements can be retrieved by calling rvSdpMsgGetNumOfMediaGroup().

RETURN VALUES

Returns the requested Media Group number.

rvSdpMediaGroupAttrGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpMediaGroupAttr* value or empty string if the value is legal.

SYNTAX

```
const char* rvSdpMediaGroupAttrGetBadSyntax(
    const RvSdpMediaGroupAttr*    mediaGroupAttr);
```

PARAMETERS

mediaGroupAttr

A pointer to the *RvSdpMediaGroupAttr*.

RETURN VALUES

Returns the bad syntax value.

rvSdpMsgGetNumOfMediaGroup()

DESCRIPTION

Gets the number of the *RvSdpMediaGroupAttr* set in the SDP message context.

SYNTAX

```
RvSize_t rvSdpMsgGetNumOfMediaGroup(
    const RvSdpMsg*    msg);
```

PARAMETERS

Msg

A pointer to an *RvSdpMsg* object.

RETURN VALUES

Returns the number of Media Group special attributes set in the *RvSdpMsg*.

rvSdpMsgGetFirstMediaGroup()

DESCRIPTION

Returns the first *RvSdpMediaGroupAttr* defined in the *RvSdpMsg*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpMediaGroupAttr* rvSdpMsgGetFirstMediaGroup(
    RvSdpMsg*          msg,
    RvSdpListIter*     iter);
```

PARAMETERS

msg

A pointer to an *RvSdpMsg* object.

Iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMsgGetNextMediaGroup() calls.

RETURN VALUES

Returns a pointer to the *RvSdpMediaGroupAttr*, or a NULL pointer if there are no Media Group objects in the *RvSdpMsg*.

rvSdpMsgGetNextMediaGroup()

DESCRIPTION

Returns the next *RvSdpMediaGroupAttr* defined in the *RvSdpMsg*. The next *RvSdpMediaGroupAttr* is defined based on the *RvSdpListIter* state.

SYNTAX

```
RvSdpMediaGroupAttr* rvSdpMsgGetNextMediaGroup(
    RvSdpListIter*    iter);
```

PARAMETERS

Iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMsgGetFirstMediaGroup() or rvSdpMsgGetNextMediaGroup() functions.

RETURN VALUES

Returns a pointer to the *RvSdpMediaGroupAttr*, or a NULL pointer if there are no more *RvSdpMediaGroupAttr* objects defined in the *RvSdpMsg*.

21

PRECONDITION ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpPreconditionAttr* objects. The *RvSdpPreconditionAttr* type represents the precondition attribute ('a=curr:' & 'a=des' & 'a=conf') fields of an *RvSdpMediaDescr*. This attribute can appear only in the context of an *RvSdpMediaDescr.* These functions can be used only if the RV_SDP_PRECONDITIONS_ATTR compilation flag (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpPreconditionAttrIsBadSyntax()
- rvSdpMsgAddBadSyntaxPrecondition()

rvSdpPreconditionAttrIsBadSyntax()

DESCRIPTION

Indicates whether or not an *RvSdpPreconditionAttr* is proprietary formatted.

SYNTAX

```
RvBool rvSdpPreconditionAttrIsBadSyntax (
    RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns RV_TRUE if the field contains bad syntax, or RV_FALSE if it does not.

rvSdpMsgAddBadSyntaxPrecondition()

DESCRIPTION

Adds a new proprietary-formatted *RvSdpPreconditionAttr* at the *RvSdpMediaDescr* level.

SYNTAX

```
RvSdpPreconditionAttr * rvSdpMsgAddBadSyntaxPrecondition (
    RvSdpMediaDescr    * descr,
    const char*        badSyn);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

badSyn

The proprietary-formatted value to be set.

RETURN VALUES

Returns a pointer to the newly-created *RvSdpPreconditionAttr* if the function succeeds, or a NULL pointer if it fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpPreconditionAttrGetPrecondTypeStr()
- rvSdpPreconditionAttrGetPrecondType()
- rvSdpPreconditionAttrSetPrecondTypeStr()
- rvSdpPreconditionAttrSetPrecondType()
- rvSdpPreconditionAttrGetStatusStrTypeStr()
- rvSdpPreconditionAttrGetStatus()
- rvSdpPreconditionAttrSetStatusStr()
- rvSdpPreconditionAttrSetStatus()
- rvSdpPreconditionAttrGetStrengthTagStr()
- rvSdpPreconditionAttrGetStrengthTag()
- rvSdpPreconditionAttrSetStrengthTagStr()
- rvSdpPreconditionAttrSetStrengthTag()
- rvSdpPreconditionAttrGetDirectionTagStr()
- rvSdpPreconditionAttrGetDirectionTag()
- rvSdpPreconditionAttrSetDirectionTagStr()
- rvSdpPreconditionAttrSetDirectionTag()
- rvSdpMediaDescrGetNumOfPrecondition()
- rvSdpMediaDescrGetFirstPrecondition()
- rvSdpMediaDescrGetNextPrecondition()
- rvSdpMediaDescrGetPrecondition()
- rvSdpPreconditionAttrGetBadSyntax()

rvSdpPreconditionAttrGetPrecondTypeStr()

DESCRIPTION

Gets the precondition type value of the *RvSdpPreconditionAttr* as a string ("QoS").

SYNTAX

```
const char* rvSdpPreconditionAttrGetPrecondTypeStr(
    const RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as a string.

rvSdpPreconditionAttrGetPrecondType()

DESCRIPTION

Gets the precondition type value of the *RvSdpPreconditionAttr* as an enumeration ("QoS").

SYNTAX

```
const RvSdpPreconditionType
rvSdpPreconditionAttrGetPrecondType(
    const RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as an enumeration.

rvSdpPreconditionAttrSetPrecondTypeStr()

DESCRIPTION

Sets the precondition type value of the *RvSdpPreconditionAttr* as a string.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetPrecondTypeStr(
    RvSdpPreconditionAttr *    precondAttr,
    const char *               precondTypeStr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

precondTypeStr

The new type value of the *RvSdpPreconditionAttr* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrSetPrecondType()

DESCRIPTION

Sets the precondition type value of the *RvSdpPreconditionAttr* as an enumeration.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetPrecondType(
    RvSdpPreconditionAttr *    precondAttr,
    RvSdpPreconditionType      ePrecondType);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

ePrecondType

The new value of the *RvSdpPreconditionAttr* type as an enumeration.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrGetStatusStrTypeStr()

DESCRIPTION

Gets the precondition status value of the *RvSdpPreconditionAttr* as a string ("e2e"/"remote"/"local").

SYNTAX

```
const char* rvSdpPreconditionAttrGetStatusStr(
    const RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as a string.

rvSdpPreconditionAttrGetStatus()

DESCRIPTION

Gets the precondition status value of the *RvSdpPreconditionAttr* as an enumeration ("e2e"/"remote"/"local").

SYNTAX

```
const RvSdpPreconditionStatusType
rvSdpPreconditionAttrGetStatus(
    const RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as an enumeration.

rvSdpPreconditionAttrSetStatusStr()

DESCRIPTION

Sets the precondition status value of the *RvSdpPreconditionAttr* as a string ("e2e"/"remote"/"local").

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetStatusStr(
    RvSdpPreconditionAttr *    precondAttr,
    const char *               precondStatusStr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

precondStatusStr

The new status value of the *RvSdpPreconditionAttr* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrSetStatus()

DESCRIPTION

Sets the precondition type value of the *RvSdpPreconditionAttr* as an enumeration.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetStatus(
    RvSdpPreconditionAttr *    precondAttr,
    RvSdpPreconditionType      ePrecondStatus);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

ePrecondStatus

The new value of the *RvSdpPreconditionAttr* status as an enumeration.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrGetStrengthTagStr()

DESCRIPTION

Gets the precondition strength value of the *RvSdpPreconditionAttr* as a string ("mandatory"/"optional"/"none"/"failure"/"unknown").

SYNTAX

```
const char* rvSdpPreconditionAttrGetStrengthTagStr(
    const RvSdpPreconditionAttr* precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as a string.

rvSdpPreconditionAttrGetStrengthTag()

DESCRIPTION

Gets the precondition strength value of the *RvSdpPreconditionAttr* as an enumeration ("mandatory"/"optional"/"none"/"failure"/"unknown").

SYNTAX

```
const RvSdpPreconditionStrengthTag
rvSdpPreconditionAttrGetStrengthTag(
    const RvSdpPreconditionAttr* precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as an enumeration.

rvSdpPreconditionAttrSetStrengthTagStr()

DESCRIPTION

Sets the precondition type value of the *RvSdpPreconditionAttr* as a string.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetStrengthTagStr(
    RvSdpPreconditionAttr *    precondAttr,
    const char *               strengthStr)
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

strengthStr

The new strength value of the *RvSdpPreconditionAttr* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrSetStrengthTag()

DESCRIPTION

Sets the precondition type value of the *RvSdpPreconditionAttr* as an enumeration.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetStrengthTag(
    RvSdpPreconditionAttr *    precondAttr,
    RvSdpPreconditionType      eStrength);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

ePrecondType

The new value of the *RvSdpPreconditionAttr* strength as an enumeration.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrGetDirectionTagStr()

DESCRIPTION

Gets the precondition direction value of the *RvSdpPreconditionAttr* as a string ("none"/"send"/"recv"/"sendrecv").

SYNTAX

```
const char* rvSdpPreconditionAttrGetDirectionTagStr(
    const RvSdpPreconditionAttr* precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as a string.

rvSdpPreconditionAttrGetDirectionTag()

DESCRIPTION

Gets the precondition direction value of the *RvSdpPreconditionAttr* as an enumeration ("none"/"send"/"recv"/"sendrecv").

SYNTAX

```
const RvSdpPreconditionDirectionTag
rvSdpPreconditionAttrGetDirectionTag(
    const RvSdpPreconditionAttr* precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the requested field of the *RvSdpPreconditionAttr* as an enumeration.

rvSdpPreconditionAttrSetDirectionTagStr()

DESCRIPTION

Sets the precondition direction value of the *RvSdpPreconditionAttr* as a string ("none"/"send"/"recv"/"sendrecv").

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetDirectionTagStr(
    RvSdpPreconditionAttr *    precondAttr,
    const char *               directionStr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

precondStatusStr

The new direction value of the *RvSdpPreconditionAttr* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpPreconditionAttrSetDirectionTag()

DESCRIPTION

Sets the precondition direction value of the *RvSdpPreconditionAttr* as an enumeration.

SYNTAX

```
RvSdpStatus rvSdpPreconditionAttrSetDirectionTag(
    RvSdpPreconditionAttr *          precondAttr,
    RvSdpPreconditionDirectionTag    eDirection);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

eDirection

The new value of the *RvSdpPreconditionAttr* direction as an enumeration.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetNumOfPrecondition()

DESCRIPTION

Gets the number of the *RvSdpPreconditionAttr* set in the SDP media level. Counts the preconditions with the same *ePrecondName*. To count all preconditions set in the *RvSdpMediaDescr*, use ePrecondName = RV_SDP_PRECOND_NAME_ALL.

SYNTAX

```
RvSize_t rvSdpMediaDescrGetNumOfPrecondition(
    RvSdpMediaDescr     * descr,
    RvSdpPrecondName    ePrecondName,
    RvSdpListIter       * iter);
```

PARAMETERS

descr

A pointer to an *RvSdpMediaDescr*.

ePrecondName

The precondition name value of the *RvSdpPreconditionAttr* as an enumeration (can be "curr"/"des"/"conf").

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMediaDescrGetNextPrecondition() calls.

RETURN VALUES

Returns the number of precondition attributes set in the *RvSdpMsg* (only preconditions with the same name as that in *ePrecondName* are counted).

rvSdpMediaDescrGetFirstPrecondition()

DESCRIPTION

Returns the first *RvSdpPreconditionAttr* defined in the *RvSdpMediaDescr* with the same name as defined in *ePrecondName*. This function also sets an *RvSdpListIter* for further use.

SYNTAX

```
RvSdpPreconditionAttr* rvSdpMediaDescrGetFirstPrecondition(
    RvSdpMediaDescr*    descr,
    RvSdpPrecondName    ePrecondName,
    RvSdpListIter       * iter);
```

PARAMETERS

descr

A pointer to an *RvSdpMediaDescr*.

ePrecondName

The precondition name value of the *RvSdpPreconditionAttr* as an enumeration (can be "curr"/"des"/"conf").

iter

A pointer to the *RvSdpListIter* to be used for further rvSdpMediaDescrGetNextPrecondition() calls.

RETURN VALUES

Returns a pointer to the first *RvSdpPreconditionAttr* object in the media level, or a NULL pointer if there are no precondition objects in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetNextPrecondition()

DESCRIPTION

Returns the next *RvSdpPreconditionAttr* defined in the *RvSdpMediaDescr* with the same name as that defined in *ePrecondName*. The definition of the next *RvSdpPreconditionAttr* is based on the *RvSdpListIter* state.

SYNTAX

```
RvSdpPreconditionAttr* rvSdpMediaDescrGetNextPrecondition(
    RvSdpPrecondName    ePrecondName,
    RvSdpListIter*      iter);
```

PARAMETERS

ePrecondName

The precondition name value of the *RvSdpPreconditionAttr* as an enumeration to be found (can be "curr"/"des"/"conf").

Iter

A pointer to the *RvSdpListIter* that was set or modified by a previous, successful call to the rvSdpMediaDescrGetFirstPrecondition() or rvSdpMediaDescrGetNextPrecondition() functions.

RETURN VALUES

Returns a pointer to the *RvSdpPreconditionAttr*,or a NULL pointer if no more *RvSdpPreconditionAttr* objects have been defined in the *RvSdpMediaDescr*.

rvSdpMediaDescrGetPrecondition()

DESCRIPTION

Gets the precondition special attribute by index.

SYNTAX

```
RvSdpPreconditionAttr* rvSdpMediaDescrGetPrecondition(
    RvSdpMediaDescr     * descr,
    RvSdpPrecondName    ePrecondName,
    RvSize_t            index);
```

PARAMETERS

descr

A pointer to an *RvSdpMediaDescr* object.

ePrecondName

The precondition name value of the *RvSdpPreconditionAttr* as an enumeration (can be "curr"/"des"/"conf").

index

The index, which should start at zero (0) and must be smaller than the number of elements with the same name value in the precondition list. The number of elements in the list can be retrieved by calling rvSdpMediaDescrGetNumOfPrecondition().

RETURN VALUES

Returns the requested precondition attribute if it exists, or NULL if it does not.

rvSdpPreconditionAttrGetBadSyntax()

DESCRIPTION

Gets a proprietary-formatted *RvSdpPreconditionAttr* value or an empty string if the value is legal.

SYNTAX

```
const char* rvSdpPreconditionAttrGetBadSyntax(
    const RvSdpPreconditionAttr*    precondAttr);
```

PARAMETERS

precondAttr

A pointer to the *RvSdpPreconditionAttr*.

RETURN VALUES

Returns the bad syntax value.

22

MSRP ATTRIBUTE FUNCTIONS

WHAT'S IN THIS CHAPTER

The *RvSdpAttribute* type represents the precondition attribute ("a=accept-types:", "a=accept-wrapped-types", "a=max-size", and "a=path"). These attributes can appear only in the context of a *RvSdpMediaDescr*. Each attribute can appear only once per *RvSdpMediaDescr*. The functions described in this section can be used only if the RV_SDP_MSRP_ATTR compilation flag (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpMediaDescrDestroyAcceptTypesAttr()
- rvSdpMediaDescrDestroyWrappedAcceptTypesAttr()
- rvSdpMediaDescrDestroyMaxSizeAttr()
- rvSdpMediaDescrDestroyPathAttr()

rvSdpMediaDescrGetAcceptTypesVal()

DESCRIPTION

Gets the accept types special attribute value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetAcceptTypesVal(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested field of the *RvSdpAttribute* as a string.

rvSdpMediaDescrSetAcceptTypesVal()

DESCRIPTION

Sets the accept types special attribute value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetAcceptTypesVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new accept types value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetAcceptWrappedTypesVal()

DESCRIPTION

Gets the accept wrapped types special attribute value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetAcceptWrappedTypesVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested value as a string.

rvSdpMediaDescrSetAcceptWrappedTypesVal()

DESCRIPTION

Sets the accept wrapped types special attribute value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetAcceptWrappedTypesVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new accept wrapped types value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetMaxSizeVal()

DESCRIPTION

Gets the max size value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetMaxSizeVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested max size value as a string.

rvSdpMediaDescrSetMaxSizeVal()

DESCRIPTION

Sets the max size value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetMaxSizeVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new max size value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetPathVal()

DESCRIPTION

Gets the path value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetPathVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested field as a string.

rvSdpMediaDescrSetPathVal()

DESCRIPTION

Sets the path value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetPathVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new path value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns the requested field of the *RvSdpAttribute* as an enumeration.

rvSdpMediaDescrDestroyAcceptTypesAttr()

DESCRIPTION

Removes the accept types special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyAcceptTypesAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyWrappedAcceptTypesAttr()

DESCRIPTION

Removes the accept wrapped types special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyWrappedAcceptTypesAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyMaxSizeAttr()

DESCRIPTION

Removes the max size special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyMaxSizeAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyPathAttr()

DESCRIPTION

Removes the path special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyPathAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpMediaDescrGetAcceptTypesVal()
- rvSdpMediaDescrSetAcceptTypesVal()
- rvSdpMediaDescrGetAcceptWrappedTypesVal()
- rvSdpMediaDescrSetAcceptWrappedTypesVal()
- rvSdpMediaDescrGetMaxSizeVal()
- rvSdpMediaDescrSetMaxSizeVal()
- rvSdpMediaDescrGetPathVal()
- rvSdpMediaDescrSetPathVal()

rvSdpMediaDescrGetAcceptTypesVal()

DESCRIPTION

Gets the accept types special attribute value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetAcceptTypesVal(
    RvSdpMediaDescr*    descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested field of the *RvSdpAttribute* as a string.

rvSdpMediaDescrSetAcceptTypesVal()

DESCRIPTION

Sets the accept types special attribute value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetAcceptTypesVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new accept types value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetAcceptWrappedTypesVal()

DESCRIPTION

Gets the accept wrapped types special attribute value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetAcceptWrappedTypesVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested value as a string.

rvSdpMediaDescrSetAcceptWrappedTypesVal()

DESCRIPTION

Sets the accept wrapped types special attribute value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetAcceptWrappedTypesVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new accept wrapped types value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetMaxSizeVal()

DESCRIPTION

Gets the max size value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetMaxSizeVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested max size value as a string.

rvSdpMediaDescrSetMaxSizeVal()

DESCRIPTION

Sets the max size value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetMaxSizeVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new max size value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

rvSdpMediaDescrGetPathVal()

DESCRIPTION

Gets the path value as a string.

SYNTAX

```
const char* rvSdpMediaDescrGetPathVal(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

Returns the requested field as a string.

rvSdpMediaDescrSetPathVal()

DESCRIPTION

Sets the path value as a string.

SYNTAX

```
RvSdpStatus rvSdpMediaDescrSetPathVal(
    RvSdpMediaDescr*    descr,
    const char          * val);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

val

The new path value of the *RvSdpAttribute* as a string.

RETURN VALUES

Returns the requested field of the *RvSdpAttribute* as an enumeration.

rvSdpMediaDescrDestroyAcceptTypesAttr()

DESCRIPTION

Removes the accept types special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyAcceptTypesAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyWrappedAcceptTypesAttr()

DESCRIPTION

Removes the accept wrapped types special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyWrappedAcceptTypesAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyMaxSizeAttr()

DESCRIPTION

Removes the max size special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyMaxSizeAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

rvSdpMediaDescrDestroyPathAttr()

DESCRIPTION

Removes the path special attribute from the media level.

SYNTAX

```
void rvSdpMediaDescrDestroyPathAttr(
    RvSdpMediaDescr* descr);
```

PARAMETERS

descr

A pointer to the *RvSdpMediaDescr.*

RETURN VALUES

None.

23

OTHER FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpOther* objects. The *RvSdpOther* Type represents a SDP line with proprietary tag, or free text SDP line. These functions can be used only if the RV_SDP_CHECK_BAD_SYNTAX compilation switch (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpOtherConstruct() **Obsolete**
- rvSdpOtherConstructA() **Obsolete**
- rvSdpOtherConstructCopy() **Obsolete**
- rvSdpOtherConstructCopyA() **Obsolete**
- rvSdpOtherDestruct()
- rvSdpOtherCopy()
- rvSdpOtherGetTag()
- rvSdpOtherSetTag()
- rvSdpOtherGetValue()
- rvSdpOtherSetValue()

rvSdpOtherConstruct()

DESCRIPTION

Constructs an *RvSdpOther*.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstruct(
    RvSdpOther*    oth,
    const char     tag,
    const char*    value);
```

PARAMETERS

oth

A pointer to a valid *RvSdpOther*.

tag

The tag letter of the line.

value

The text of the line.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructA()

DESCRIPTION

Constructs an *RvSdpOther*.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructA(
    RvSdpOther*    oth,
    const char     tag,
    const char*    value,
    RvAlloc*       a);
```

PARAMETERS

oth

A pointer to a valid *RvSdpOther*.

tag

The tag letter of the line.

value

The text of the line.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructCopy()

DESCRIPTION

Constructs an *RvSdpOther* and copies the values from a source *RvSdpOther* field.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructCopy(
    RvSdpOther*          dest,
    const RvSdpOther*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpOther* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpOther*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructCopyA()

DESCRIPTION

Constructs an *RvSdpOther* and copies the values from a source *RvSdpOther* field. **This function is obsolete.** Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructCopyA(
    RvSdpOther*          dest,
    const RvSdpOther*    src,
    RvAlloc*             a);
```

PARAMETERS

dest

A pointer to the *RvSdpOther* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpOther*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherDestruct()

DESCRIPTION

Destructs an *RvSdpOther*.

SYNTAX

```
void rvSdpOtherDestruct(
    RvSdpOther    *oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

RETURN VALUES

None.

rvSdpOtherCopy()

DESCRIPTION

Copies the values from a source *RvSdpOther* to a destination *RvSdpOther.*

SYNTAX

```
RvSdpOther* rvSdpOtherCopy(
    RvSdpOther*          dest,
    const RvSdpOther*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpOther.* This parameter must point to a constructed *RvSdpOther.*

src

The *RvSdpOther.*

RETURN VALUES

Returns a pointer to the destination *RvSdpOther*, or NULL if the function fails.

rvSdpOtherGetTag()

DESCRIPTION

Gets the tag letter.

SYNTAX

```
char rvSdpOtherGetTag(
    const RvSdpOther*    oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

RETURN VALUES

Returns the tag letter.

rvSdpOtherSetTag()

DESCRIPTION

Sets the tag letter of the line.

SYNTAX

```
void rvSdpOtherSetTag(
    RvSdpOther*    oth,
    const char     tag);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

tag

The tag letter of the line.

RETURN VALUES

None.

rvSdpOtherGetValue()

DESCRIPTION

Gets the line value of an *RvSdpOther* after the equal sign (=).

SYNTAX

```
const char* rvSdpOtherGetValue(
    const RvSdpOther*    oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther.*

RETURN VALUES

Returns the line value of the SDP after the equal sign (=).

rvSdpOtherSetValue()

DESCRIPTION

Sets the line value of the SDP after the equal sign (=).

SYNTAX

```
RvSdpStatus rvSdpOtherSetValue(
    RvSdpOther*    oth,
    const char     *value);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

value

The line value of the SDP.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpOtherGetTag()
- rvSdpOtherSetTag()
- rvSdpOtherGetValue()
- rvSdpOtherSetValue()

rvSdpOtherConstruct()

DESCRIPTION

Constructs an *RvSdpOther*.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstruct(
    RvSdpOther*    oth,
    const char     tag,
    const char*    value);
```

PARAMETERS

oth

A pointer to a valid *RvSdpOther*.

tag

The tag letter of the line.

value

The text of the line.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructA()

DESCRIPTION

Constructs an *RvSdpOther*.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructA(
    RvSdpOther*    oth,
    const char     tag,
    const char*    value,
    RvAlloc*       a);
```

PARAMETERS

oth

A pointer to a valid *RvSdpOther*.

tag

The tag letter of the line.

value

The text of the line.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructCopy()

DESCRIPTION

Constructs an *RvSdpOther* and copies the values from a source *RvSdpOther* field.

This function is obsolete. Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructCopy(
    RvSdpOther*          dest,
    const RvSdpOther*    src);
```

PARAMETERS

dest

A pointer to the *RvSdpOther* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpOther*.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherConstructCopyA()

DESCRIPTION

Constructs an *RvSdpOther* and copies the values from a source *RvSdpOther* field. **This function is obsolete.** Please use rvSdpMsgAddOther() or rvSdpMediaDescrAddOther() instead.

SYNTAX

```
RvSdpOther* rvSdpOtherConstructCopyA(
    RvSdpOther*          dest,
    const RvSdpOther*    src,
    RvAlloc*             a);
```

PARAMETERS

dest

A pointer to the *RvSdpOther* to be constructed. This parameter must point to valid memory.

src

The source *RvSdpOther*.

a

The *RvAlloc* to be used for memory allocations. If the *RvAlloc* is NULL, the default *RvAlloc* is used.

RETURN VALUES

Returns a pointer to the constructed *RvSdpOther*, or NULL if the function fails.

rvSdpOtherDestruct()

DESCRIPTION

Destructs an *RvSdpOther*.

SYNTAX

```
void rvSdpOtherDestruct(
    RvSdpOther    *oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

RETURN VALUES

None.

rvSdpOtherCopy()

DESCRIPTION

Copies the values from a source *RvSdpOther* to a destination *RvSdpOther*.

SYNTAX

```
RvSdpOther* rvSdpOtherCopy(
    RvSdpOther*          dest,
    const RvSdpOther*    src);
```

PARAMETERS

dest

A pointer to the destination *RvSdpOther*. This parameter must point to a constructed *RvSdpOther*.

src

The *RvSdpOther*.

RETURN VALUES

Returns a pointer to the destination *RvSdpOther*, or NULL if the function fails.

rvSdpOtherGetTag()

DESCRIPTION

Gets the tag letter.

SYNTAX

```
char rvSdpOtherGetTag(
    const RvSdpOther*    oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

RETURN VALUES

Returns the tag letter.

rvSdpOtherSetTag()

DESCRIPTION

Sets the tag letter of the line.

SYNTAX

```
void rvSdpOtherSetTag(
    RvSdpOther*    oth,
    const char     tag);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

tag

The tag letter of the line.

RETURN VALUES

None.

rvSdpOtherGetValue()

DESCRIPTION

Gets the line value of an *RvSdpOther* after the equal sign (=).

SYNTAX

```
const char* rvSdpOtherGetValue(
    const RvSdpOther*    oth);
```

PARAMETERS

oth

A pointer to the *RvSdpOther.*

RETURN VALUES

Returns the line value of the SDP after the equal sign (=).

rvSdpOtherSetValue()

DESCRIPTION

Sets the line value of the SDP after the equal sign (=).

SYNTAX

```
RvSdpStatus rvSdpOtherSetValue(
    RvSdpOther*    oth,
    const char     *value);
```

PARAMETERS

oth

A pointer to the *RvSdpOther*.

value

The line value of the SDP.

RETURN VALUES

Returns RV_SDPSTATUS_OK if the function succeeds, or an error code if the function fails.

24

BAD SYNTAX FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to operate on *RvSdpBadSyntax* objects. *RvSdpBadSyntax* is used to hold a standard SDP line that contains proprietary information. Each *RvSdpMsg* and *RvSdpMediaDescr* contain a list of these objects. This object is used to directly access the non-standard lines (or lines with syntax errors). These functions can be used only if the RV_SDP_CHECK_BAD_SYNTAX compilation switch (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

GET/SET FUNCTIONS

This section describes the following functions:

- rvSdpBadSyntaxGetField()
- rvSdpBadSyntaxGetFieldType()

rvSdpBadSyntaxGetField()

DESCRIPTION

Gets the SDP object contained in an *RvSdpBadSyntax*.

SYNTAX

```
RvSdpGenericFieldPtr rvSdpBadSyntaxGetField(
    const RvSdpBadSyntax*    badS);
```

PARAMETERS

badS

A pointer to the *RvSdpBadSyntax*.

RETURN VALUES

Returns a handle to the contained object. This handle can be cast to the contained SDP object according to its type. The type can be retrieved using the rvSdpBadSyntaxGetFieldType() function.

rvSdpBadSyntaxGetFieldType()

DESCRIPTION

Gets the type of SDP object contained in an *RvSdpBadSyntax*. The user can use this function to know which object is wrapped inside the *RvSdpBadSyntax*.

SYNTAX

```
RvSdpFieldTypes rvSdpBadSyntaxGetFieldType(
    const RvSdpBadSyntax*    badS);
```

PARAMETERS

badS

A pointer to the *RvSdpBadSyntax*.

RETURN VALUES

Returns the type of the wrapped object. This type is mapped by the RvSdpFieldTypes enumeration.

25

SDP OBJECTS REPARSE FUNCTIONS

WHAT'S IN THIS CHAPTER

This section contains the functions used to reparse bad syntax SDP objects. These functions can be used only if the RV_SDP_ENABLE_REPARSE compilation switch (defined in the *rvsdpconfig.h* file) is enabled.

Included in this section are:

CONTROL FUNCTIONS

This section describes the following functions:

- rvSdpOriginReparse()
- rvSdpUriReparse()
- rvSdpEmailReparse()
- rvSdpPhoneReparse()
- rvSdpConnectionReparse()
- rvSdpBandwidthReparse()
- rvSdpSessionTimeReparse()
- rvSdpRepeatIntReparse()
- rvSdpKeyReparse()
- rvSdpMediaDescrReparse()
- rvSdpRtpMapReparse()

rvSdpOriginReparse()

DESCRIPTION

Performs a reparse of an *RvSdpOrigin*. The bad syntax of the *RvSdpOrigin* is used for reparse. If the reparse succeeds, the origin field becomes a valid SDP object. Use rvSdpOriginGetBadSyntax() and rvSdpOriginSetBadSyntax() to access the bad syntax field of the *RvSdpOrigin*. If the reparse fails, the *RvSdpOrigin* remains unchanged.

SYNTAX

```
RvSdpOrigin* rvSdpOriginReparse(
    RvSdpMsg*            msg,
    RvSdpOrigin*         c,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to *RvSdpMsg* where the *RvSdpOrigin* is contained.

c

A pointer to the reparsed *RvSdpOrigin*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpOrigin* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpUriReparse()

DESCRIPTION

Performs a reparse of the URI field of the message.

SYNTAX

```
const char* rvSdpUriReparse(
    RvSdpMsg*            msg,
    const char*          u,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the URI is contained.

u

The URI line for reparse.

len

Thus parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to valid URI field in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpEmailReparse()

DESCRIPTION

Performs a reparse of an *RvSdpEmail*. The bad syntax of the *RvSdpEmail* is used for reparse. If the reparse succeeds, the *RvSdpEmail* field becomes a valid SDP object. Use rvSdpEmailGetBadSyntax() and rvSdpEmailSetBadSyntax() to access the bad syntax field of the *RvSdpEmail*. If the reparse fails, the object remains unchanged.

SYNTAX

```
RvSdpEmail* rvSdpEmailReparse(
    RvSdpMsg*            msg,
    RvSdpEmail*          c,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpEmail* is contained.

c

A pointer to the reparsed *RvSdpEmail*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpEmail* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpPhoneReparse()

DESCRIPTION

Performs a reparse of an *RvSdpPhone*. The bad syntax of the *RvSdpPhone* is used for reparse. If the reparse succeeds, the *RvSdpPhone* field becomes a valid SDP object. Use rvSdpPhoneGetBadSyntax() and rvSdpPhoneDestruct() to access the bad syntax field of the *RvSdpPhone*. If the reparse fails, the *RvSdpPhone* remains unchanged.

SYNTAX

```
RvSdpPhone* rvSdpPhoneReparse(
    RvSdpMsg*            msg,
    RvSdpPhone*          c,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpPhone* is contained.

c

A pointer to the reparsed *RvSdpPhone*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpPhone* in case of a successful reparse. Otherwise a NULL pointer is returned.

rvSdpConnectionReparse()

DESCRIPTION

Performs a reparse of an *RvSdpConnection*. The bad syntax of the *RvSdpConnection* is used for reparse. If the reparse succeeds, the *RvSdpConnection* field becomes a valid SDP object. Use rvSdpConnectionGetBadSyntax() and rvSdpConnectionSetBadSyntax() to access the bad syntax field of the *RvSdpConnection*. If the reparse fails, the *RvSdpConnection* remains unchanged.

SYNTAX

```
RvSdpConnection* rvSdpConnectionReparse(
    RvSdpMsg*            msg,
    RvSdpConnection*     c,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpConnection* is contained.

c

A pointer to the reparsed *RvSdpConnection*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpConnection* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpBandwidthReparse()

DESCRIPTION

Performs a reparse of an *RvSdpBandwidth*. The bad syntax of the *RvSdpBandwidth* is used for reparse. If the reparse succeeds, the *RvSdpBandwidth* field becomes a valid SDP object. Use rvSdpBandwidthGetBadSyntax() and rvSdpBandwidthSetBadSyntax() to access bad syntax field of the *RvSdpBandwidth*. If the reparse fails, the *RvSdpBandwidth* remains unchanged.

SYNTAX

```
RvSdpBandwidth* rvSdpBandwidthReparse(
    RvSdpMsg*            msg,
    RvSdpBandwidth*      b,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpBandwidth* is contained.

c

A pointer to the reparsed *RvSdpBandwidth*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpBandwidth* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpSessionTimeReparse()

DESCRIPTION

Performs a reparse of an *RvSdpSessionTime*. The bad syntax of the *RvSdpSessionTime* is used for reparse. If the reparse succeeds, the *RvSdpSessionTime* field becomes a valid SDP object. Use rvSdpSessionTimeGetBadSyntax() and rvSdpSessionTimeSetBadSyntax() to access bad syntax field of the *RvSdpSessionTime*. If the reparse fails, the *RvSdpSessionTime* remains unchanged.

SYNTAX

```
RvSdpSessionTime* rvSdpSessionTimeReparse(
    RvSdpMsg*            msg,
    RvSdpSessionTime*    b,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpSessionTime* is contained.

c

A pointer to the reparsed *RvSdpSessionTime*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpSessionTime* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpRepeatIntReparse()

DESCRIPTION

Performs a reparse of an *RvSdpRepeatInterval*. The bad syntax of the *RvSdpRepeatInterval* is used for reparse. If the reparse succeeds, the *RvSdpRepeatInterval* field becomes a valid SDP object. Use rvSdpRepeatIntervalGetBadSyntax() and rvSdpRepeatIntervalSetBadSyntax() to access bad syntax field of the *RvSdpRepeatInterval*. If the reparse fails, the *RvSdpRepeatInterval* remains unchanged.

SYNTAX

```
RvSdpRepeatInterval* rvSdpRepeatIntReparse(
    RvSdpMsg*               msg,
    RvSdpRepeatInterval*    b,
    int*                    len,
    RvSdpParseStatus*       stat,
    RvAlloc*                a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpRepeatInterval* is contained.

c

A pointer to the reparsed *RvSdpRepeatInterval*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpRepeatInterval* in case of a successful reparse. Otherwise a NULL pointer is returned.

rvSdpKeyReparse()

DESCRIPTION

Performs a reparse of an RvSdpKey. The bad syntax of the *RvSdpKey* is used for reparse. If the reparse succeeds, the *RvSdpKey* field becomes a valid SDP object. Use rvSdpKeyGetBadSyntax() and rvSdpKeySetBadSyntax() to access the bad syntax field of the RvSdpKey. If the reparse fails, the *RvSdpKey* remains unchanged.

SYNTAX

```
RvSdpKey* rvSdpKeyReparse(
    RvSdpMsg*            msg,
    RvSdpKey*            b,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpKey* is contained.

c

A pointer to the reparsed *RvSdpKey.*

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpKey* in case of a successful reparse. Otherwise, a NULL pointer is returned.

rvSdpMediaDescrReparse()

DESCRIPTION

Performs a reparse of an *RvSdpMediaDescr*. The bad syntax of the *RvSdpMediaDescr* is used for reparse. If the reparse succeeds, the *RvSdpMediaDescr* field becomes a valid SDP object. Use rvSdpMediaDescrGetBadSyntaxValue() and rvSdpMediaDescrSetBadSyntax() to access the bad syntax field of the *RvSdpMediaDescr*. If the reparse fails, the *RvSdpMediaDescr* remains unchanged.

SYNTAX

```
RvSdpMediaDescr* rvSdpMediaDescrReparse(
    RvSdpMsg*           msg,
    RvSdpMediaDescr*    b,
    int*                len,
    RvSdpParseStatus*   stat,
    RvAlloc*            a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpMediaDescr* is contained.

c

A pointer to the reparsed *RvSdpMediaDescr*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpMediaDescr* in case of a successful reparse. Otherwise a NULL pointer is returned.

rvSdpRtpMapReparse()

DESCRIPTION

Performs a reparse of an *RvSdpRtpMap*. The bad syntax of the *RvSdpRtpMap* is used for reparse. If the reparse succeeds, the *RvSdpRtpMap* field becomes a valid SDP object. Use rvSdpRtpMapGetBadSyntax() and rvSdpRtpMapSetBadSyntax() to access the bad syntax field of the *RvSdpRtpMap*. If the reparse fails, the *RvSdpRtpMap* remains unchanged.

SYNTAX

```
RvSdpRtpMap* rvSdpRtpMapReparse(
    RvSdpMsg*            msg,
    RvSdpRtpMap*         b,
    int*                 len,
    RvSdpParseStatus*    stat,
    RvAlloc*             a);
```

PARAMETERS

msg

A pointer to the *RvSdpMsg* where the *RvSdpRtpMap* is contained.

c

A pointer to the reparsed *RvSdpRtpMap*.

len

This parameter is not used.

stat

The reparse status which is set at the end of the reparse process.

a

This parameter is not used.

RETURN VALUES

Returns a pointer to the *RvSdpRtpMap* in case of a successful reparse. Otherwise, a NULL pointer is returned.

26

ENUMERATIONS

WHAT'S IN THIS CHAPTER

This section contains the Enumerated Type definitions of the SDP API. These type definitions are contained in the *rvsdpsymb.h* file.

Included in this section are:

ENUMERATED TYPE DEFINITIONS

This section describes the following Enumerated Type definitions:

- RvSdpParseStatus
- RvSdpEncrMethod
- RvSdpTimeUnit
- RvSdpConnectionMode
- RvSdpProtocol
- RvSdpNetType
- RvSdpAddrType
- RvSdpMediaType
- RvSdpStatus
- RvSdpFieldTypes
- RvSdpKeyMgmtPrtclType
- RvSdpGroupSemanticsType
- RvSdpPrecondName
- RvSdpPreconditionType
- RvSdpPreconditionStatusType
- RvSdpPreconditionStrengthTag
- RvSdpPreconditionDirectionTag

RvSdpParseStatus

DESCRIPTION

Describes the status of a completed parse operation.

SYNTAX

```
RV_SDPPARSER_STOP_ZERO
RV_SDPPARSER_STOP_BLANKLINE
RV_SDPPARSER_STOP_DOTLINE
RV_SDPPARSER_STOP_CLOSEBRACE
RV_SDPPARSER_STOP_ALLOCFAIL
RV_SDPPARSER_STOP_ERROR
```

PARAMETERS

RV_SDPPARSER_STOP_ZERO

Parsing stopped successfully at a null character.

RV_SDPPARSER_STOP_BLANKLINE

Parsing stopped successfully at a blank character or a new line.

RV_SDPPARSER_STOP_DOTLINE

Parsing stopped successfully at a new line starting with a dot.

RV_SDPPARSER_STOP_CLOSEBRACE

Parsing stopped successfully at a closing brace.

RV_SDPPARSER_STOP_ALLOCFAIL

Parsing failed due to a memory allocation error.

RV_SDPPARSER_STOP_ERROR

Parsing failed due to a syntax error.

RvSdpEncrMethod

DESCRIPTION

Describes the type of encryption method.

SYNTAX

```
RV_SDPENCRMTHD_CLEAR
RV_SDPENCRMTHD_BASE64
RV_SDPENCRMTHD_URI
RV_SDPENCRMTHD_PROMPT
RV_SDPENCRMTHD_KEY
```

PARAMETERS

RV_SDPENCRMTHD_CLEAR

Send in clear text (no encryption).

RV_SDPENCRMTHD_BASE64

Send base64 encoded. (Includes characters that are prohibited in SDP).

RV_SDPENCRMTHD_URI

A Universal Resource Identifier as used by WWW clients is included in this key field.

RV_SDPENCRMTHD_PROMPT

No key is included in this SDP description. The user should be prompted for the key.

RV_SDPENCRMTHD_KEY

The encryption key type.

RvSdpTimeUnit

DESCRIPTION

Describes the type of time unit.

SYNTAX

```
RV_SDPTIMETYPE_DAY
RV_SDPTIMETYPE_HOUR
RV_SDPTIMETYPE_MONTH
RV_SDPTIMETYPE_SECOND
```

PARAMETERS

RV_SDPTIMETYPE_DAY

The time given is in units of days.

RV_SDPTIMETYPE_HOUR

The time given is in units of hours.

RV_SDPTIMETYPE_MONTH

The time given is in units of months.

RV_SDPTIMETYPE_SECOND

The time given is in units of seconds.

RvSdpConnectionMode

DESCRIPTION

Describes the type of connecting mode.

SYNTAX

```
RV_SDPCONNECTMODE_NOTSET
RV_SDPCONNECTMODE_SENDONLY
RV_SDPCONNECTMODE_RECVONLY
RV_SDPCONNECTMODE_SENDRECV
RV_SDPCONNECTMODE_INACTIVE
```

PARAMETERS

RV_SDPCONNECTMODE_NOTSET

The connection mode is not set.

RV_SDPCONNECTMODE_SENDONLY

The connection mode is send only.

RV_SDPCONNECTMODE_RECVONLY

The connection mode is receive only.

RV_SDPCONNECTMODE_SENDRECV

The connection mode is send/receive.

RV_SDPCONNECTMODE_INACTIVE

The connection mode is inactive.

RvSdpProtocol

DESCRIPTION

Describes the type of protocol.

SYNTAX

```
RV_SDPPROTOCOL_RTP
RV_SDPPROTOCOL_LOCAL
RV_SDPPROTOCOL_ATM
RV_SDPPROTOCOL_UDP
RV_SDPPROTOCOL_AAL1ATMF
RV_SDPPROTOCOL_AAL1ITU
RV_SDPPROTOCOL_AAL1CUSTOM
RV_SDPPROTOCOL_AAL2ATMF
RV_SDPPROTOCOL_AAL2ITU
RV_SDPPROTOCOL_AAL2CUSTOM
RV_SDPPROTOCOL_AAL5ATMF
RV_SDPPROTOCOL_AAL5ITU
RV_SDPPROTOCOL_AAL5CUSTOM
RV_SDPPROTOCOL_H323C
```

PARAMETERS

RV_SDPPROTOCOL_RTP

Use RTP.

RV_SDPPROTOCOL_ATM

Use ATM.

RV_SDPPROTOCOL_UDP

Use UDP.

RV_SDPPROTOCOL_AAL1ATMF

Use ATM Adaption Layer 1, ATMF.

RV_SDPPROTOCOL_AAL1ITU

Use ATM Adaption Layer 1, ITU.

RV_SDPPROTOCOL_AAL1CUSTOM

Use ATM Adaption Layer 1, custom.

RV_SDPPROTOCOL_AAL2ATMF

Use ATM Adaption Layer 2, ATMF.

RV_SDPPROTOCOL_AAL2ITU

Use ATM Adaption Layer 2, ITU.

RV_SDPPROTOCOL_AAL2CUSTOM

Use ATM Adaption Layer 2, custom.

RV_SDPPROTOCOL_AAL5ATMF

Use ATM Adaption Layer 5, ATMF.

RV_SDPPROTOCOL_AAL5ITU

Use ATM Adaption Layer 5, ITU.

RV_SDPPROTOCOL_AAL5CUSTOM

Use ATM Adaption Layer 5, custom.

RV_SDPPROTOCOL_H323C

Use H.323 Annex C.

RvSdpNetType

DESCRIPTION

Describes the type of network.

SYNTAX

```
RV_SDPNETTYPE_IN
RV_SDPNETTYPE_ATM
RV_SDPNETTYPE_LOCAL
RV_SDPNETTYPE_OTHER
RV_SDPNETTYPE_TN
RV_SDPNETTYPE_ANY
```

PARAMETERS

RV_SDPNETTYPE_IN

Network type is Internet.

RV_SDPNETTYPE_ATM

Network type is ATM.

RV_SDPNETTYPE_LOCAL

Network type is local.

RV_SDPNETTYPE_OTHER

Network type is other.

RV_SDPNETTYPE_TN

Network type is TN.

RV_SDPNETTYPE_ANY

Network type is any.

RvSdpAddrType

DESCRIPTION

Describes the type of address.

SYNTAX

```
RV_SDPADDRTYPE_IP4
RV_SDPADDRTYPE_IP6
RV_SDPADDRTYPE_ENDPOINT
RV_SDPADDRTYPE_NSAP
RV_SDPADDRTYPE_E164
RV_SDPADDRTYPE_GWID
RV_SDPADDRTYPE_ALIAS
RV_SDPADDRTYPE_RFC2543
```

PARAMETERS

RV_SDPADDRTYPE_IP4

Address is an IP version 4 address.

RV_SDPADDRTYPE_IP6

Address is an IP version 6 address.

RV_SDPADDRTYPE_ENDPOINT

Address is an endpoint.

RV_SDPADDRTYPE_NSAP

Address is an NSAP address.

RV_SDPADDRTYPE_E164

Address is E.164 (SMDS, Frame Relay, ATM).

RV_SDPADDRTYPE_GWID

Address type is GWID.

RV_SDPADDRTYPE_ALIAS

Address is an alias.

RV_SDPADDRTYPE_RFC2543

Address type is specified by RFC2543 (SIP).

RvSdpMediaType

DESCRIPTION

Describes the type of media.

SYNTAX

```
RV_SDPMEDIATYPE_AUDIO
RV_SDPMEDIATYPE_NAS
RV_SDPMEDIATYPE_VIDEO
RV_SDPMEDIATYPE_APP
RV_SDPMEDIATYPE_DATA
RV_SDPMEDIATYPE_IMAGE
RV_SDPMEDIATYPE_CONTROL
```

PARAMETERS

RV_SDPMEDIATYPE_AUDIO

Media type is audio.

RV_SDPMEDIATYPE_NAS

Media type is NAS.

RV_SDPMEDIATYPE_VIDEO

Media type is video.

RV_SDPMEDIATYPE_APP

Media type is application specific.

RV_SDPMEDIATYPE_DATA

Media type is data.

RV_SDPMEDIATYPE_IMAGE

Media type is image.

RV_SDPMEDIATYPE_CONTROL

Media type is control.

RvSdpStatus

DESCRIPTION

Describes the return status of a function.

SYNTAX

```
RV_SDPSTATUS_OK
RV_SDPSTATUS_ENCODEFAILBUF
RV_SDPSTATUS_ALLOCFAIL
RV_SDPSTATUS_PARSEFAIL
```

PARAMETERS

RV_SDPSTATUS_OK

Operation completed successfully.

RV_SDPSTATUS_ENCODEFAILBUF

Encoding failed due to insufficient buffer space.

RV_SDPSTATUS_ALLOCFAIL

Operation failed due to a memory allocation error.

RV_SDPSTATUS_PARSEFAIL

Operation failed due to a parsing error.

RvSdpFieldTypes

DESCRIPTION

Describes the object type.

SYNTAX

```
SDP_FIELDTYPES_ORIGIN,
SDP_FIELDTYPES_INFORMATION,
SDP_FIELDTYPES_SESSION,
SDP_FIELDTYPES_URI,
SDP_FIELDTYPES_EMAIL,
SDP_FIELDTYPES_PHONE,
SDP_FIELDTYPES_BANDWIDTH,
SDP_FIELDTYPES_VERSION,
SDP_FIELDTYPES_TIME,
SDP_FIELDTYPES_KEY,
SDP_FIELDTYPES_ATTRIBUTE,
SDP_FIELDTYPES_CONNECTION,
SDP_FIELDTYPES_REPEAT,
SDP_FIELDTYPES_MEDIA,
SDP_FIELDTYPES_RTP_MAP,
SDP_FIELDTYPES_KEY_MGMT,
SDP_FIELDTYPES_CRYPTO,
SDP_FIELDTYPES_CONNECTION_MODE,
SDP_FIELDTYPES_FRAMERATE,
SDP_FIELDTYPES_FMTP,
SDP_FIELDTYPE_MEDIA_ID,
SDP_FIELDTYPE_MEDIA_GROUP,
SDP_FIELDTYPE_PRECONDITION,
SDP_FIELDTYPES_UNKNOWN_TAG,
SDP_FIELDTYPES_BAD_FIELD
```

PARAMETERS

SDP_FIELDTYPES_ORIGIN

The object is *RvSdpOrigin*.

SDP_FIELDTYPES_INFORMATION

The object is *RvSdpInformation*.

SDP_FIELDTYPES_SESSION

The object is *RvSdpSessionTime*.

SDP_FIELDTYPES_URI

The object is URI.

SDP_FIELDTYPES_EMAIL

The object is *RvSdpEmail*.

SDP_FIELDTYPES_PHONE

The object is *RvSdpPhone*.

SDP_FIELDTYPES_BANDWIDTH

The object is *RvSdpBandwidth*.

SDP_FIELDTYPES_TIME

For internal use of the Stack.

SDP_FIELDTYPES_KEY

The object is *RvSdpKey*.

SDP_FIELDTYPES_ATTRIBUTE

For internal use of the Stack.

SDP_FIELDTYPES_CONNECTION

The object is *RvSdpConnection*.

SDP_FIELDTYPES_REPEAT

For internal use of the SDP Stack.

SDP_FIELDTYPES_MEDIA

The object is *RvSdpMediaDescr*.

SDP_FIELDTYPES_RTP_MAP

The object is *RvSdpRtpMap*.

SDP_FIELDTYPES_KEY_MGMT

The object is *RvSdpKeyMgmtAttr*.

SDP_FIELDTYPES_CRYPTO

The object is *RvSdpCryptoAttr*.

SDP_FIELDTYPES_CONNECTION_MODE

For connection mode special attributes.

SDP_FIELDTYPES_FRAMERATE

For framerate special attributes.

SDP_FIELDTYPES_FMTP

For *fmtp* special attributes.

SDP_FIELDTYPE_MEDIA_ID

For *mid* special attributes.

SDP_FIELDTYPE_MEDIA_GROUP

The object is *RvSdMediaGroupAttr*.

SDP_FIELDTYPE_PRECONDITION

The object is *RvSdPreconditionpAttr*.

SDP_FIELDTYPES_UNKNOWN_TAG

For internal use of the SDP Stack.

SDP_FIELDTYPES_BAD_FIELD

Error indication.

RvSdpKeyMgmtPrtclType

DESCRIPTION

Describes the type of key management attribute protocol ID.

SYNTAX

```
RV_SDPKEYMGMT_NOTSET
RV_SDPKEYMGMT_MIKEY
RV_SDPKEYMGMT_UNKNOWN
```

PARAMETERS

RV_SDPKEYMGMT_NOTSET

The protocol ID is not set.

RV_SDPKEYMGMT_MIKEY

The key management protocol is “mikey”.

RV_SDPKEYMGMT_UNKNOWN

The proprietary (unregistered) protocol is used for key management. Use the rvSdpKeyMgmtGetPrtclIdTxt() function to fetch the textual value of the protocol ID.

RvSdpGroupSemanticsType

DESCRIPTION

Describes the semantics type of the Media Group attribute.

SYNTAX

```
RV_SDP_GROUP_SEMANTICS_NOTSET
RV_SDP_GROUP_SEMANTICS_LS
RV_SDP_GROUP_SEMANTICS_FID
RV_SDP_GROUP_SEMANTICS_SRF
RV_SDP_GROUP_SEMANTICS_UNKNOWN
```

PARAMETERS

RV_ SDP_GROUP_SEMANTICS _NOTSET

The semantics value is not set.

RV_ SDP_GROUP_SEMANTICS _LS

The semantics value is “LS”.

RV_SDP_GROUP_SEMANTICS_FID

The semantics value is “FID”.

RV_SDP_GROUP_SEMANTICS_SRF

The semantics value is “SRF”.

RV_ SDP_GROUP_SEMANTICS _UNKNOWN

The proprietary (unregistered) semantics is used for the Media Group.

RvSdpPrecondName

DESCRIPTION

Describes the name of the precondition attribute.

SYNTAX

```
RV_SDP_PRECOND_NAME_CURRENT_ATTR
RV_SDP_PRECOND_NAME_DESIRED_ATTR
RV_SDP_PRECOND_NAME_CONFIRMED_ATTR
RV_SDP_PRECOND_NAME_ALL
RV_SDP_PRECOND_NAME_UNKNOWN
```

PARAMETERS

RV_ SDP_ PRECOND_NAME_CURRENT_ATTR

The precondition name is “curr”.

RV_ SDP_ PRECOND_NAME_ DESIRED _ATTR

The precondition name is “des”.

RV_ SDP_ PRECOND_NAME_ CONFIRMED _ATTR

The precondition name is “conf”.

RV_ SDP_ PRECOND_NAME ALL

All names.

RV_ SDP_ PRECOND_ NAME_UNKNOWN

The proprietary (unregistered) name is used for the precondition attribute.

RvSdpPreconditionType

DESCRIPTION

Describes the type of precondition attribute (the first string in the value part).

SYNTAX

```
RV_SDP_PRECOND_TYPE_NOTSET
RV_SDP_PRECOND_TYPE_QOS
RV_SDP_PRECOND_TYPE_UNKNOWN
```

PARAMETERS

RV_SDP_PRECOND_TYPE_NOTSET

The precondition type was not set.

RV_SDP_PRECOND_TYPE_QOS

The precondition type is Quality of Service (QoS).

RV_SDP_PRECOND_TYPE_UNKNOWN

The precondition type is Unknown.

RvSdpPreconditionStatusType

DESCRIPTION

Describes the status type of the precondition attribute (the second string parameter in the value part (for “curr” and “conf” names), and the third parameter in the value part (for “des” name).

SYNTAX

```
RV_SDP_PRECOND_STATUS_TYPE_NOTSET
RV_SDP_PRECOND_STATUS_TYPE_E2E
RV_SDP_PRECOND_STATUS_TYPE_LOCAL
RV_SDP_PRECOND_STATUS_TYPE_REMOTE
RV_SDP_PRECOND_STATUS_TYPE_UNKNOWN
```

PARAMETERS

RV_SDP_PRECOND_STATUS_TYPE_NOTSET

The precondition status was not set.

RV_SDP_PRECOND_STATUS_TYPE_E2E

The precondition status type is End to End (E2E).

RV_SDP_PRECOND_STATUS_TYPE_LOCAL

The precondition status type is Local.

RV_SDP_PRECOND_STATUS_TYPE_REMOTE

The precondition status type is Remote.

RV_SDP_PRECOND_STATUS_TYPE_UNKNOWN

The precondition status type is Unknown.

RvSdpPreconditionStrengthTag

DESCRIPTION

Describes the Strength tag of the precondition attribute (the second string parameter in the value part, only in the “desire” precondition attribute name).

SYNTAX

```
RV_SDP_PRECOND_STRENGTH_NOTSET
RV_SDP_PRECOND_STRENGTH_NONE
RV_SDP_PRECOND_STRENGTH_OPTIONAL
RV_SDP_PRECOND_STRENGTH_MANDATORY
RV_SDP_PRECOND_STRENGTH_FAILURE
RV_SDP_PRECOND_STRENGTH_UNKNOWN
```

PARAMETERS

RV_SDP_PRECOND_STRENGTH_NOTSET

The precondition strength tag was not set.

RV_SDP_PRECOND_STRENGTH_NONE

The precondition strength tag is None.

RV_SDP_PRECOND_STRENGTH_OPTIONAL

The precondition strength tag is Optional.

RV_SDP_PRECOND_STRENGTH_MANDATORY

The precondition strength tag is Mandatory.

RV_SDP_PRECOND_STRENGTH_FAILURE

The precondition strength tag is Failure.

RV_SDP_PRECOND_STRENGTH_UNKNOWN

The precondition strength tag is Unknown.

RvSdpPreconditionDirectionTag

DESCRIPTION

Describes the direction tag of the precondition attribute (the final string parameter in the value part).

SYNTAX

```
RV_SDP_PRECOND_DIRECTION_NOTSET
RV_SDP_PRECOND_DIRECTION_NONE
RV_SDP_PRECOND_DIRECTION_SEND
RV_SDP_PRECOND_DIRECTION_RECV
RV_SDP_PRECOND_DIRECTION_SENDRECV
RV_SDP_PRECOND_DIRECTION_UNKNOWN
```

PARAMETERS

RV_SDP_PRECOND_DIRECTION_NOTSET

The precondition direction tag was not set.

RV_SDP_PRECOND_DIRECTION_NONE

The precondition direction tag is None.

RV_SDP_PRECOND_DIRECTION_SEND

The precondition direction tag is Send.

RV_SDP_PRECOND_DIRECTION_RECV

The precondition direction tag is Recv.

RV_SDP_PRECOND_DIRECTION_SENDRECV

The precondition direction tag is SendRecv.

RV_SDP_PRECOND_DIRECTION_UNKNOWN

The precondition direction tag is Unknown.

INDEX

R

S