M5T SDP SAFE v1.7 - API Reference

Proprietary & Confidential

Legal Information

Document Build Date

This document was built on: May 26, 2010

Table of Contents

1 - Introduction

Welcome to the M5T SDP SAFE API Reference. This document provides quick and simple references to the usage of the SDP SAFE parser.

All modifications to this API Reference are tracked through CRs (Change Requests) and are documented in the release notes associated with this product.

M5T SDP SAFE v1.7 still supports a number of APIs that are deprecated. A deprecated API means that it is accessible in this version of the software, but will be eventually removed, so its use is discouraged.

2 - MXD_SDP_H264_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Source Code: #define MXD_SDP_H264_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Enables the support for parsing the H.264 FMTP attributes according to RFC 3984.

Enabling this define adds a new class CSdpFmtpH264 and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

3 - MXD_SDP_H263_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Source Code: #define MXD_SDP_H263_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Enables the support for parsing the H.263 FMTP attributes according to draft-ietf-avt-rfc2429-bis-09.txt.

Enabling this define adds a new class CSdpFmtpH263 and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

4 - MXD_SDP_MP4V_ES_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Source Code: #define MXD_SDP_MP4V_ES_FMTP_ATTRIBUTE_ENABLE_SUPPORT

Enables the support for parsing the MPEG-4 FMTP visual attributes according to RFC 3016.

Enabling this define adds a new class CSdpFmtpMp4Ves and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

5 - Components Overview

This sections describes the content of the M5T SDP SAFE. The documentation is divided into subsections, each of which describes one specific part of the M5T SDP SAFE.

6 - Compile-Time Configuration

6.1 - Configuring SDP SAFE with "PreSdpParserCfg.h"

SDP SAFE comes with the file "Config/SdpParserCfg.h" which defines many compilation configuration options and values used specifically by the SDP Parser. Generally, these values will need updating for the specific application being developed with SDP SAFE.

To update these default values, you must create the "PreSdpParserCfg.h" file with the updated configuration options for your application. "PreSdpParserCfg.h" is always included first by "SdpParserCfg.h" to retrieve application specific configurations, and then the default configuration options found in "Config/SdpParserCfg.h" are applied for all items that were not configured by the application.

"PreSdpParserCfg.h" is not packaged with SDP SAFE and must be created for the specific application being developed. This file must simply be placed somewhere in the compiler search path to permit the retrieval of the application specific configuration options by SDP SAFE.

6.2 - Configuration Macros

Macros

Macro	Description
MXD_SDP_B2BUA_CONNECTOR_ENABLE_SUPPORT (see page 6)	Enables the support for the B2BUA connector.
MXD_SDP_ENABLE_AMR_FMTP_ATTRIBUTE (see page 7)	Enables the support for AMR FMTP attributes.
MXD_SDP_ENABLE_G7221_FMTP_ATTRIBUTE (see page 7)	Enables the support for G.722.1 FMTP attributes.
MXD_SDP_ENABLE_ICE_LITE_WITHOUT_CC_NEGOTIATION (see page 7)	Enables the address negotiation using ICE lite.
MXD_SDP_ENABLE_ILBC_FMTP_ATTRIBUTE (see page 8)	Enables the support for iLBC FMTP attribute.
MXD_SDP_ENABLE_ISAC_FMTP_ATTRIBUTE (see page 8)	Enables the support for iSAC FMTP attributes.
MXD_SDP_ENABLE_KEY_MANAGEMENT_MIKEY_ATTRIBUTE (see page 8)	Enables the integration of M5T MIKEY key management.
MXD_SDP_ENABLE_MPTIME (see page 9)	Enables parsing and serialization of the attribute mptime.
MXD_SDP_ENABLE_REDUNDANCY_FMTP_ATTRIBUTE (see page 9)	Enables the support for redundancy FMTP attribute.
MXD_SDP_ENABLE_SRTP_SUPPORT (see page 9)	Enables the support of SRTP.
MXD_SDP_ENABLE_T38_SUPPORT (see page 10)	Enables the support of T.38.
MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE (see page 10)	Enables the support of telephone event FMTP attribute.
MXD_SDP_ICE_ENABLE_SUPPORT (see page 10)	Enables the support for the ICE attributes.
MXD_SDP_KEY_MANAGEMENT_NEGOTIATION_ENABLE_SUPPORT (see page 10)	Enables the support for key management attribute negotiation.
MXD_SDP_SILENCE_SUPPRESSION_ENABLE_SUPPORT (see page 11)	Enables the support of silenceSupp media level attribute.
MXD_SDP_SILENCE_SUPPRESSION_INDICATION_ABSENCE_MEANS_DISABLED (see page 11)	When defined, indicates that silence suppression is to be considered disabled when the "annex=" fmtp parameter is not specified whithin a session description.
MXD_SDP_SUPPORT_MISSING_MEDIA_LINE_IN_ANSWER (see page 11)	Enables the support of missing media lines in SDP answers.
MXD_SDP_SUPPORT_NON_COMPLIANT_SENDRECV_ANSWER (see page 12)	Enables the support of non-compliant send/receive attribute in SDP answers.
MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_ENUM (see page 12)	Enables the extension of the compression algorithm enum array.
MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_NAME (see page 12)	Enables the extension of the compression algorithm string array.
MXD_SDPCAPSMGR_USER_EXTEND_RTPMAP_ARRAY (see page 12)	Enables the extension of the RTP Map array.

6.2.1 - MXD_SDP_B2BUA_CONNECTOR_ENABLE_SUPPORT Macro

Enables the support for the B2BUA connector.

C++

```
#define MXD_SDP_B2BUA_CONNECTOR_ENABLE_SUPPORT
```

Description

Enables the support for the B2BUA connector. This helper class implements the necessary functionality required to allow connecting at the media level two UAs that previously had their media routed through a third-party entity. This handles only the on / off part. The rest of the parsing is up to the application.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpB2BUaConnector

6.2.2 - MXD_SDP_ENABLE_AMR_FMTP_ATTRIBUTE Macro

Enables the support for AMR FMTP attributes.

C++

```
#define MXD_SDP_ENABLE_AMR_FMTP_ATTRIBUTE
```

Description

Enables the support for parsing the AMR FMTP attributes.

Enabling this define adds a new class CSdpFmtpAMR and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.3 - MXD_SDP_ENABLE_G7221_FMTP_ATTRIBUTE Macro

Enables the support for G.722.1 FMTP attributes.

C++

```
#define MXD_SDP_ENABLE_G7221_FMTP_ATTRIBUTE
```

Description

Enables the support for parsing the G.722.1 FMTP attributes.

Enabling this define adds a new class CSdpFmtpG7221 and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.4 - MXD_SDP_ENABLE_ICE_LITE_WITHOUT_CC_NEGOTIATION Macro

Enables the address negotiation using ICE lite.

C++

```
#define MXD_SDP_ENABLE_ICE_LITE_WITHOUT_CC_NEGOTIATION
```

Description

Enables negotiation using ICE candidates without connectivity check.

The application is responsible to add ICE candidates into each media of the capabilities manager corresponding to the addresses of its RTP and RTCP streams.

When sending an offer, the capabilities manager automatically takes care of:

- Choosing one candidate for each component (RTP and RTCP) to be the default destination. If IPv4 is supported, the IPv4 candidates have priority over IPv6.
- Setting the IP address of the RTP candidate in the c line and the port of the RTP candidate in the m line.
- Setting the IP address of the RTCP candidate in the RTCP attribute.
- Adding the ice-lite, ice-ufrag and ice-pwd attributes if not already present.

When generating an answer, the capabilities manager automatically takes care of:

- Checking if there is the ice-lite attribute, if there are ICE candidates, and if all default destinations match an ICE candidate. If not, the answer is processed as normal RFC 3264 procedures.
- Selecting the default destination for each component based on local priority of the candidates. Only one candidate per component is

put into the answer.

- Setting the IP address of the RTP candidate in the c line and the port of the RTP candidate in the m line.
- Setting the IP address of the RTCP candidate in the RTCP attribute.
- Adding the ice-lite, ice-ufrag and ice-pwd attributes if not already present.
- If candidate IP versions are not compatible, the GenerateAnswer method fails.

When validating an answer, the capabilities manager automatically takes care of:

- Checking if there is the ice-lite attribute, if there are ICE candidates, and if all default destinations match an ICE candidate. If not, the answer is processed as normal RFC 3264 procedures.
- Validating if the received ICE candidate IP versions are compatible with the offered ones. If they are not compatible, the VerifyAnswer method fails.

Location

Define this in PreSdpParserCfg.h.

6.2.5 - MXD_SDP_ENABLE_ILBC_FMTP_ATTRIBUTE Macro

Enables the support for iLBC FMTP attribute.

C++

```
#define MXD_SDP_ENABLE_ILBC_FMTP_ATTRIBUTE
```

Description

Enables the support for parsing the iLBC FMTP attribute.

Enabling this define adds a new class CSdpFmtpIlbc. When this macro is defined, the capabilities manager will make sure to include the local fmtp attribute if it is available. The iLBC ftmp attribute advertises the local capability and does not require negotiation. This means that the incoming and outgoing streams can use different modes.

iLBC is a public codec initially published by GIPS (Global IP Solutions) and is defined by RFC 3951 and RFC 3952.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.6 - MXD_SDP_ENABLE_ISAC_FMTP_ATTRIBUTE Macro

Enables the support for iSAC FMTP attributes.

C++

```
#define MXD_SDP_ENABLE_ISAC_FMTP_ATTRIBUTE
```

Description

Enables the support for parsing the iSAC FMTP attributes.

Enabling this define adds a new class CSdpFmtpIsac and the necessary negotiation in the CSdpCapabilitiesMgr (see page 25).

iSAC is a proprietary codec from GIPS (Global IP Solutions) and is defined by this draft: http://tools.ietf.org/html/draft-legrand-rtp-isac-02

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.7 - MXD_SDP_ENABLE_KEY_MANAGEMENT_MIKEY_ATTRIBUTE Macro

Enables the integration of M5T MIKEY key management.

C++

```
#define MXD_SDP_ENABLE_KEY_MANAGEMENT_MIKEY_ATTRIBUTE
```

Description

Enables the integration of M5T MIKEY key management in SDP. Negotiation is done according to RFC 4567. In this mode, only MIKEY key-mgmt attributes are supported and all other key-mgmt attribute, whether supported or not, are ignored. When using this mode, users must configure key management using the MIKEY class.

When an application sets up a MIKEY key-mgmt attribute in this mode with the corresponding CSdpKeyManagementParameterMikey (see page 283), the SDP capabilities manager does the following in client mode:

- Generates a MIKEY message according to the configured information for each media, base 64 encodes it, and adds it to the SDP packet.
- When the answer is received, it decodes the MIKEY answer, parses the MIKEY message, and then sets the needed data in the corresponding CSdpKeyManagementParameterMikey (see page 283).

In server mode:

- When generating an answer, it matches the first MIKEY key-mgmt with the first locally supported one. It parses this message and configures the CSdpKeyManagementParameterMikey (see page 283).
- When the answer is returned, it generates a MIKEY response or error message according to the configured parameters.

MXD_SDP_ENABLE_KEY_MANAGEMENT_MIKEY_ATTRIBUTE MUST be defined for this define to be available.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.8 - MXD_SDP_ENABLE_MPTIME Macro

Enables parsing and serialization of the attribute mptime.

C++

```
#define MXD_SDP_ENABLE_MPTIME
```

Description

When defined, the attribute a=mptime is parsed. The value can then be used, modified and serialized.

Location

Define this in PreSdpParserCfg.h.

6.2.9 - MXD_SDP_ENABLE_REDUNDANCY_FMTP_ATTRIBUTE Macro

Enables the support for redundancy FMTP attribute.

C++

```
#define MXD_SDP_ENABLE_REDUNDANCY_FMTP_ATTRIBUTE
```

Description

Enables the support for redundancy FMTP attribute in the SDP capabilities manager.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.10 - MXD_SDP_ENABLE_SRTP_SUPPORT Macro

Enables the support of SRTP.

C++

```
#define MXD_SDP_ENABLE_SRTP_SUPPORT
```

Description

This switch is used to control whether or not the SDP parser recognizes the parameters specific to SRTP when parsing a media line in an SDP packet. You can enable this switch if your application wants to support the SRTP additions to SDP. This enables the compilation of a few classes and thus increases the size of the build.

Location

Define this in PreSdpParserCfg.h if SRTP parsing classes are required.

6.2.11 - MXD_SDP_ENABLE_T38_SUPPORT Macro

Enables the support of T.38.

C++

```
#define MXD_SDP_ENABLE_T38_SUPPORT
```

Description

This switch is used to control whether or not the SDP parser recognizes the parameters specific to T.38 when parsing a media line in an SDP packet. You can enable this switch if your application wants to support the T.38 fax protocol. This enables the compilation of a few classes and thus increases the size of the build.

Location

Define this in PreSdpParserCfg.h if T.38 parsing classes are required.

6.2.12 - MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE Macro

Enables the support of telephone event FMTP attribute.

C++

```
#define MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE
```

Description

Enables the support for telephone event FMTP attribute in the SDP capabilities manager.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.13 - MXD_SDP_ICE_ENABLE_SUPPORT Macro

Enables the support for the ICE attributes.

C++

```
#define MXD_SDP_ICE_ENABLE_SUPPORT
```

Description

Enables the support for the ICE attributes (parsing and negotiation).

Location

Define this in PreSdpParserCfg.h or in your makefile.

6.2.14 - MXD_SDP_KEY_MANAGEMENT_NEGOTIATION_ENABLE_SUPPORT Macro

Enables the support for key management attribute negotiation.

C++

```
#define MXD_SDP_KEY_MANAGEMENT_NEGOTIATION_ENABLE_SUPPORT
```

Description

Enables the support for key management attribute negotiation in the SDP capabilities manager. Negotiation is done according to RFC 4567 by using the first key management attribute supported both locally and by the peer. The local string configured in the CSdpFieldAttributeKeyMgmt (see page 134) is sent to the peer.

Location

Define this in PreSdpParserCfg.h or in your makefile.

See Also

CSdpCapabilitiesMgr (see page 25)

6.2.15 - MXD_SDP_SILENCE_SUPPRESSION_ENABLE_SUPPORT Macro

Enables the support of silenceSupp media level attribute.

C++

```
#define MXD_SDP_SILENCE_SUPPRESSION_ENABLE_SUPPORT
```

Description

When defined, the SDP parser supports the silenceSupp media level attribute. The capabilities manager is also able to negotiate the silenceSupp attribute (fmtp).

Note that silenceSupp is intepreted as a media level attribute only (not valid at the session level). http: www.iana.org/assignments/sdp-parameters specifies that silenceSupp is not considered in one of three attribute categories: att-field (session level) att-field (both session and media level) att-field (media level only)

but is considered as unknown: att-field (unknown level)

The IETF itself does not consider the silenceSupp as a media or session level attribute, and it is a matter of implementation decision.

Location

Define this in PreSdpParserCfg.h.

6.2.16 - MXD_SDP_SILENCE_SUPPRESSION_INDICATION_ABSENCE_MEANS_DISABLED Macro

When defined, indicates that silence suppression is to be considered disabled when the "annex=" fmtp parameter is not specified whithin a session description.

C++

```
#define MXD_SDP_SILENCE_SUPPRESSION_INDICATION_ABSENCE_MEANS_DISABLED
```

Description

When defined, the capabilities manager will consider that VAD is not enabled if the "annex=" format attribute is absent. The default behaviour is to consider VAD as enabled, as per RFC 3555.

This macro only impacts the behaviour if the "annex=" format attribute is absent. If the format attribute is present, its value will have precedence. Since this macro also affects the interpretation of the "index=" format attribute absence from the local caps, it is recommended to always explicitely specify if VAD is enabled or not when using this macro.

Location

Define this in PreSdpParserCfg.h.

6.2.17 - MXD_SDP_SUPPORT_MISSING_MEDIA_LINE_IN_ANSWER Macro

Enables the support of missing media lines in SDP answers.

C++

```
#define MXD_SDP_SUPPORT_MISSING_MEDIA_LINE_IN_ANSWER
```

Description

This switch allows the VerifyAnswer method to match media lines even if the peer removed some media lines in the response.

Location

Define this in PreSdpParserCfg.h or in your makefile.

6.2.18 - MXD_SDP_SUPPORT_NON_COMPLIANT_SENDRECV_ANSWER Macro

Enables the support of non-compliant send/receive attribute in SDP answers.

C++

```
#define MXD_SDP_SUPPORT_NON_COMPLIANT_SENDRECV_ANSWER
```

Description

This switch allows the VerifyAnswer method to accept answers containing streams with an attribute of a=sendrecv when the offer specified a=sendonly or a=recvonly.

Location

Define this in PreSdpParserCfg.h or in your makefile.

6.2.19 - MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_ENUM Macro

Enables the extension of the compression algorithm enum array.

C++

```
#define MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_ENUM
```

Description

This switch enables users to extend the CSdpParser::ERtpCompressionAlgorithm enum to support codecs that are not included in M5T's standard list. Users must define this to the enum values that would be put inside the enum declaration, each additional codec separated with a comma. When defined, this value must be terminated be a comma.

Location

Define this in PreSdpParserCfg.h or in your makefile.

Example

```
Note:To compile this sample you must either add a backslash at the end
of the first line or have everything on the same line.

#define MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_ENUM eEXTENDED1,
                                                       eEXTENDED2,
```

6.2.20 - MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_NAME Macro

Enables the extension of the compression algorithm string array.

C++

```
#define MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_NAME
```

Description

This switch enables users to extend the CSdpParser::s_aszRtpCompressionAlgorithmMap array of strings to support codecs that are not included in M5T's standard list. Users must define this to the text that would be put inside a string array, each additional codec separated with a comma. When defined, this value must be terminated by a comma.

Location

Define this in PreSdpParserCfg.h or in your makefile.

Example

```
Note:To compile this sample you must either add a backslash at the end
of the first line or have everything on the same line.

#define MXD_SDP_USER_EXTEND_COMPRESSION_ALGORITHM_NAME "ExtendedCodec1",
                                                       "ExtendedCodec2",
```

6.2.21 - MXD_SDPCAPSMGR_USER_EXTEND_RTPMAP_ARRAY Macro

Enables the extension of the RTP Map array.

C++

```
#define MXD_SDPCAPSMGR_USER_EXTEND_RTPMAP_ARRAY
```

Description

This switch enables users to extend the CSdpCapsManager::s_aRtpAlgorithmMap array, to support codecs that are not included in

M5T's standard list. This array holds the default payload number, clock rate, and number of encoding parameters associated with a codec. Users must define this to the definition of a structure instance that would be put inside an array of structures, each additional codec separated with a comma. When defined, this value must be terminated by a comma.

Warning

Care should be taken that the used payload numbers do not clash with those already set by M5T in the CSdpCapsManager::s_aRtpAlgorithmMap array.

Location

Define this in PreSdpParserCfg.h or in your makefile.

Example

{ payload number, clock rate, number of encoding parameters }

Using a number of encoding parameters less than 2 will result in it not being serialized on output as this is the SDP default value.

```
Note:To compile this sample you must either add a backslash at the end
of the first line or have everything on the same line.

  #define MXD_SDPCAPSMGR_USER_EXTEND_RTPMAP_ARRAY
          {109, 5000,   1},
          {110, 16000,  4},
```

7 - Tracing in SDP SAFE

7.1 - The M5T Tracing Mechanism

The M5T SDP SAFE integrates the M5T tracing mechanism, which uses tracing nodes to allow the application to control which part of the M5T component can output traces. See MxTraceEnableNode and MxTraceDisableNode in the M5T Framework API reference document to find out more on how to enable and disable tracing through tracing nodes.

7.2 - M5T SDP SAFE Tracing Nodes

The root node of the M5T SDP SAFE is "SdpParser". No other tracing nodes are available.

```
      /SdpParser
```

8 - Startup APIs

This section documents the application level APIs that are available from the SDP SAFE Startup package, which is located under the Startup directory.

The Startup offers a class that is responsible to initialize and terminate SDP SAFE before it can be used.

8.1 - Startup Classes

Classes

Class	Description
CSdpParserInitializer (see page 15)	This class is responsible of doing the initialization of SDP SAFE.

8.1.1 - CSdpParserInitializer Class

This class is responsible of doing the initialization of SDP SAFE.

Class Hierarchy

C++

```
class CSdpParserInitializer;
```

Description

Initializes the SDP Parser and its dependencies.

Location

Startup/CSdpParserInitializer.h

Methods

Method	Description
◆ Finalize (see page 15)	Finalizes the SDP Parser and its dependencies.
◆ Initialize (see page 15)	Initializes the SDP Parser and its dependencies.

Legend

◆	Method

8.1.1.1 - Methods

8.1.1.1.1 - CSdpParserInitializer::Finalize Method

Finalizes the SDP Parser and its dependencies.

C++

```
static void Finalize();
```

Description

Finalizes the SDP Parser and its dependencies. The application must call no method on any other classes or objects of the SDP Parser after this method is called. This method must be called only from one thread and must not be called from a different thread than the Initialize (see page 15) method was called.

8.1.1.1.2 - CSdpParserInitializer::Initialize Method

Initializes the SDP Parser and its dependencies.

C++

```
static mxt_result Initialize();
```

Returns

- resS_OK: The SDP Parser and its dependencies were properly

initialized. The parser needs to be Finalized when it is no longer needed.

- resFE_FAIL: One or more components failed to be initialized.

Description

Initializes the SDP Parser and its dependencies. This method must be called before any method on any other classes or objects of the SDP Parser. This method must be called only from one thread.

9 - SDP Management APIs

This section documents the application level APIs that are available from the SDP SAFE management package, which is located under the SdpMgmt directory.

This package offers the basic and advanced management methods for any application to easily use SDP payloads within the offer/answer model.

9.1 - SDP SAFE Management Classes

Classes

Class	Description
CSdpB2bUaConnector (see page 17)	This class represents a B2BUA connector.
CSdpCapabilitiesMgr (see page 25)	This class is used to find and manage the common capabilities between a UAC and a UAS.

9.1.1 - CSdpB2bUaConnector Class

This class represents a B2BUA connector.

Class Hierarchy

C++

```
class CSdpB2bUaConnector;
```

Description

This helper class implements the functionality required to allow connecting at the media level two UAs that previously had their media routed through a third-party entity.

This helper class is useful only for devices acting like B2B-UA and that have previously established SDP sessions with different UAs it now wants to connect together.

```
   UA-A       B2BUA       UA-B
    |           |          |
    |    RTP    |    RTP   |
    |<--------->|<-------->|
    |           |          |
    | INVITE    |          |
    |<----------|          |
    |           |          |
    |200 w/sdp  |          |
    |---------->|          |
    |           |INV w/sdp |
    |           |--------->|
    |           |          |
    |           |200 w/sdp |
    |           |<---------|
    |ACK w/sdp  |          |
    |<----------|          |
    |           | ACK      |
    |           |--------->|
    |           |          |
    |          RTP         |
    |<-------------------->|
    |           |          |
```

As per the example above, the B2BUA has already negotiated independent SDP sessions with both endpoints, but now it wants to connect them together in order to save resources. It does so by using the 3PCC mechanism. However, there are certain items that must remain coherent at the SDP level. This class is used to perform this task of keeping and showing a coherent view of the SDP session of a single endpoint. Depending on the B2BUA implementation, two instances of this class may be used back to back, or a single instance can be used by each call leg.

Location

SdpMgmt/CSdpB2bUaConnector.h

Updating an Offer

The following sequence shows how to use this class after it was initialized. It highlights the responsibility of each leg to use either UpdateOffer (see page 24) or UpdateAnswer (see page 23) when it is either receiving SDP from the endpoint or forwarding SDP to the endpoint.

```
  Client-1           B2BUA-Leg-1        B2BUA-Leg-2          Client-2
     |                   |                   |                  |
     | INVITE w/ offerA  |                   |                  |
     |------------------>|                   |                  |
     |              UpdateOffer()            |                  |
     |                   |                   |                  |
     |                   | Internal Comm w/  |                  |
     |                   |     offerA        |                  |
     |                   |------------------>|                  |
     |                   |              UpdateOffer()           |
     |                   |                   |                  |
     |                   |                   | INVITE w/ offerA'|
     |                   |                   |----------------->|
     |                   |                   |                  |
     |                   |                   |                  |
     |                   |                   |  200 w/ answerA  |
     |                   |                   |<-----------------|
     |                   |              UpdateAnswer()          |
     |                   |                   |                  |
     |                   | Internal Comm w/  |                  |
     |                   |    answerA        |                  |
     |                   |<------------------|                  |
     |              UpdateAnswer()           |                  |
     |                   |                   |                  |
     |  200 w/ answerA'  |                   |                  |
     |<------------------|                   |                  |
     |                   |                   |                  |
```

Modes of operation

Stateful operation: In stateful mode, the B2BUA is created, initialized, and uses the stateful methods to update the offers and answers. It updates both m and o lines to make sure the packet will match the SDP session it will be sent on and keeps track of the changes made to both sides.

Stateless operation: In stateless mode, the B2BUA uses only two methods: StatelessUpdateOffer (see page 23)() and StatelessUpdateAnswer (see page 22)(). These two methods basically do the same thing, which is to make sure the media lines in the new SDP matches the ones in the previously sent packets. However, it does not update the packet's o line nor does it keep track of changes.

Constructors

Constructor	Description
◆ CSdpB2bUaConnector (see page 19)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpB2bUaConnector (see page 19)	Destructor.

Legend

◆	Method
V	virtual

Methods

Method	Description
◆V CancelOffer (see page 20)	Informs that the last offer was rejected.
◆ GetUpdateTable (see page 20)	Gets the update table vector.
◆ InitializeSdpSession (see page 20)	B2BUA Initialization. First step: sets the direction of pLastOffer-pLastAnswer. SetLastOffer (see page 22)() and SetLastAnswer (see page 21)() must be called to complete the initialization.
◆ SetLastAnswer (see page 21)	B2BUA Initialization. Sets the pLastAnswer sent or received with the endpoint. SetLastOffer (see page 22)() must be also called to complete the initialization.
◆ SetLastOffer (see page 22)	B2BUA Initialization. Sets the pLastOffer sent or received with the endpoint. SetLastAnswer (see page 21)() must be also called to complete the initialization.
◆ SetMaximumMLineIndex (see page 22)	Sets the maximum m-line index in the packet.
◆ StatelessUpdateAnswer (see page 22)	Statelessly updates an answer.

◆ StatelessUpdateOffer (see page 23)	Statelessly updates an offer.
◆ V TerminateB2BUaConnector (see page 23)	Terminates the B2BUA connector operation.
◆ V UpdateAnswer (see page 23)	Updates an answer received from the endpoint or from another entity.
◆ V UpdateOffer (see page 24)	Updates an offer received from the endpoint or from another entity.

Legend

◆	Method
V	virtual

Enumerations

Enumeration	Description
EMessageToUpdateDirection (see page 25)	Message to update direction.
EOfferAnswerDirection (see page 25)	Offer answer direction

Structs

Struct	Description
SUpdateTabEntry (see page 19)	Update table structure.

9.1.1.1 - Structs

9.1.1.1.1 - CSdpB2bUaConnector::SUpdateTabEntry Struct

Update table structure.

C++

```
struct SUpdateTabEntry {
  int16_t m_nSrcIndex;
  int16_t m_nDstIndex;
  EUpdateTabOpCode m_eOpCode;
  void* m_pXtraData;
};
```

Description

Update table structure.

Members

Members	Description
`int16_t m_nSrcIndex;`	"m=" line index in the received packet.
`int16_t m_nDstIndex;`	"m=" line index in the updated packet.
`EUpdateTabOpCode m_eOpCode;`	Update operation code.
`void* m_pXtraData;`	Extra data pointer: media formats to delete, type of media announcement to add, etc.

9.1.1.2 - Constructors

9.1.1.2.1 - CSdpB2bUaConnector::CSdpB2bUaConnector Constructor

Default constructor.

C++

```
CSdpB2bUaConnector();
```

9.1.1.3 - Destructors

9.1.1.3.1 - CSdpB2bUaConnector::~CSdpB2bUaConnector Destructor

Destructor.

C++

```
virtual ~CSdpB2bUaConnector();
```

9.1.1.4 - Methods

9.1.1.4.1 - CSdpB2bUaConnector::CancelOffer Method

Informs that the last offer was rejected.

C++

```
virtual mxt_result CancelOffer();
```

Returns

- resS_OK: The offer was cancelled.

Description

This method informs the connector that there is no answer to provide to a previous offer.

This allows the connector to release resources allocated by the call to UpdateOffer (see page 24).

See Also

UpdateOffer (see page 24).

9.1.1.4.2 - CSdpB2bUaConnector::GetUpdateTable Method

Gets the update table vector.

C++

```
CVList<SUpdateTabEntry>* GetUpdateTable();
```

Returns

- The update table or NULL if there is none yet.

Description

Gets the update table currently in use.

Warning

Should only be used by advanced users as changing this table will alter the result of the next updates.

9.1.1.4.3 - InitializeSdpSession

9.1.1.4.3.1 - CSdpB2bUaConnector::InitializeSdpSession Method

B2BUA Initialization. First step: sets the direction of pLastOffer-pLastAnswer. SetLastOffer (see page 22)() and SetLastAnswer (see page 21)() must be called to complete the initialization.

C++

```
mxt_result InitializeSdpSession(IN EOfferAnswerDirection eDirection);
```

Parameters

Parameters	Description
`IN EOfferAnswerDirection eDirection`	Indicates whether the "last offer" was generated by the B2BUA or by the peer.

Returns

- resS_OK: the first step of the B2BUA initialization was completed.
- resFE_INVALID_STATE: The endpoint was already initialized.

Description

This method initializes the B2BUA connector and sets the direction of the "last" offer-answer exchanged with the endpoint. This method allows to initialize B2BUA connector in an alternative 3-steps way. After it has been called, the object is not ready to work and it remains in a provisional state waiting for the setting of lastOffer and lastAnswer.

See Also

SetLastOffer (see page 22), SetLastAnswer (see page 21).

9.1.1.4.3.2 - CSdpB2bUaConnector::InitializeSdpSession Method

Configures the SDP session currently shared between the B2BUA and an endpoint.

C++

```
virtual mxt_result InitializeSdpSession(IN TO CSdpPacket* pLastOffer, IN TO CSdpPacket* pLastAnswer, IN
EOfferAnswerDirection eDirection);
```

Parameters

Parameters	Description
`IN TO CSdpPacket* pLastOffer`	The last offer sent or received with the endpoint. The ownership is TAKEN when the function returns without errors.
`IN TO CSdpPacket* pLastAnswer`	The last answer sent or received with the endpoint. The ownership is TAKEN when the function returns without errors.
`IN EOfferAnswerDirection eDirection`	Indicates whether the offer was generated by the B2BUA or by the peer.

Returns

- resS_OK: Endpoint was initialized.
- resFE_INVALID_STATE: The endpoint was already initialized.
- resFE_INVALID_ARGUMENT: Problems were encountered with the SDP passed in parameter.
- resFE_FAIL: Internal initialization has failed.

Description

This method configures the B2BUA connector with the SDP session of the local endpoint. pLastOffer and pLastAnswer must be the last offer and answer that were exchanged with the endpoint.

Once this method has been called for the endpoint, any new offer or answer from the endpoint to another peer or from a peer to the endpoint must be provided to this object through UpdateOffer (see page 24) or UpdateAnswer (see page 23).

See Also

UpdateOffer (see page 24), UpdateAnswer (see page 23)

9.1.1.4.4 - CSdpB2bUaConnector::SetLastAnswer Method

B2BUA Initialization. Sets the pLastAnswer sent or received with the endpoint. SetLastOffer (see page 22)() must be also called to complete the initialization.

C++

```
mxt_result SetLastAnswer(IN TO CSdpPacket* pLastAnswer);
```

Parameters

Parameters	Description
`IN TO CSdpPacket* pLastAnswer`	The last answer sent or received with the endpoint. The ownership is TAKEN when the function returns without errors.

Returns

- resS_OK: Last answer was set.
- resFE_INVALID_STATE: SetLastAnswer() was not expected in the current objet state.
- resFE_INVALID_ARGUMENT: Problems were encountered with the SDP packet passed as parameter.
- resFE_FAIL: Internal initialization has failed.

Description

This method sets the last answer exchanged with the endpoint before B2BUA starts. If SetLastOffer (see page 22) was already called and if this function returns without errors, B2BUA will be in the "ready" state and it will be ready to work. Otherwise, B2BUA will remain waiting for SetLastOffer (see page 22).

See Also

InitializeSdpSession (see page 20), SetLastOffer (see page 22).

9.1.1.4.5 - CSdpB2bUaConnector::SetLastOffer Method

B2BUA Initialization. Sets the pLastOffer sent or received with the endpoint. SetLastAnswer (see page 21)() must be also called to complete the initialization.

C++

```
mxt_result SetLastOffer(IN TO CSdpPacket* pLastOffer);
```

Parameters

Parameters	Description
`IN TO CSdpPacket* pLastOffer`	The last offer sent or received with the endpoint. The ownership is TAKEN when the function returns without errors.

Returns

- resS_OK: LastOffer was set.
- resFE_INVALID_STATE: SetLastOffer() was not expected in the current objet state.
- resFE_INVALID_ARGUMENT: Problems were encountered with the SDP packet passed as parameter.
- resFE_FAIL: Internal initialization has failed.

Description

This method sets the last offer exchanged with the endpoint before B2BUA starts. If SetLastAnswer (see page 21) was already called and if this function returns without errors, B2BUA will be in the "ready" state and it will be ready to work. Otherwise, B2BUA will remain waiting for pLastAnswer.

See Also

InitializeSdpSession (see page 20), SetLastAnswer (see page 21).

9.1.1.4.6 - CSdpB2bUaConnector::SetMaximumMLineIndex Method

Sets the maximum m-line index in the packet.

C++

```
void SetMaximumMLineIndex(int16_t uMaxIndex);
```

Parameters

Parameters	Description
`int16_t uMaxIndex`	The new maximum m-line index.

Description

Sets the maximum m-line index in the packet.

Warning

Should only be used by advanced users as changing this index may have unforeseen consequences.

9.1.1.4.7 - CSdpB2bUaConnector::StatelessUpdateAnswer Method

Statelessly updates an answer.

C++

```
static mxt_result StatelessUpdateAnswer(IN const CSdpCapabilitiesMgr& rLastOffer, INOUT CSdpCapabilitiesMgr&
rNewSdpAnswer);
```

Parameters

Parameters	Description
`IN const CSdpCapabilitiesMgr& rLastOffer`	The last local offer received.
`INOUT CSdpCapabilitiesMgr& rNewSdpAnswer`	The new peer answer to update.

Returns

- resS_OK: The answer was properly updated.
- resFE_INVALID_ARGUMENT: Problems were found with the SDP answer passed in parameter.

Description

This method is used to statelessly notify the B2B UA Connector that an answer was received and may need to be updated.

This method will update the answer to make it valid with regards to the other SDP session.

The o=, v=, s=, i=, u=, e=, p=, b=, t=, r=, z=, and k= lines are not modified.

See Also

StatelessUpdateOffer (see page 23)

9.1.1.4.8 - CSdpB2bUaConnector::StatelessUpdateOffer Method

Statelessly updates an offer.

C++

```
static mxt_result StatelessUpdateOffer(IN const CSdpCapabilitiesMgr& rLastAnswer, INOUT CSdpCapabilitiesMgr&
rNewSdpOffer);
```

Parameters

Parameters	Description
`IN const CSdpCapabilitiesMgr& rLastAnswer`	The last local answer received.
`INOUT CSdpCapabilitiesMgr& rNewSdpOffer`	The new peer offer to update.

Returns

- resS_OK: The offer was properly updated.
- resFE_INVALID_ARGUMENT: Problems were found with the SDP offer passed in parameter.

Description

This method is used to statelessly notify the B2B UA Connector that an offer has been received and may need to be updated.

This method updates the offer in order to ensure that both SDP sessions are compatible and that both endpoints can exchange RTP directly.

The o=, v=, s=, i=, u=, e=, p=, b=, t=, r=, z= and k= lines are not modified.

See Also

StatelessUpdateAnswer (see page 22)

9.1.1.4.9 - CSdpB2bUaConnector::TerminateB2BUaConnector Method

Terminates the B2BUA connector operation.

C++

```
virtual mxt_result TerminateB2BUaConnector();
```

Returns

- resS_OK: Connector was terminated without errors.
- resFE_INVALID_STATE: The endpoint was already terminated.

Description

This method terminates the B2BUA connector releasing the allocated resources.

Once this method has been called for the endpoint, the B2BUA connector remains in the CLOSED state and it can no longer be used.

See Also

InitializeSdpSession (see page 20).

9.1.1.4.10 - CSdpB2bUaConnector::UpdateAnswer Method

Updates an answer received from the endpoint or from another entity.

C++

```
virtual mxt_result UpdateAnswer(INOUT CSdpPacket* pAnswer, IN EMessageToUpdateDirection eMessageDirection =
eMESSAGE_UNDEF_DIRECTION);
```

Parameters

Parameters	Description
`INOUT CSdpPacket* pAnswer`	The answer to update.
`IN EMessageToUpdateDirection eMessageDirection = eMESSAGE_UNDEF_DIRECTION`	Indicates the message direction: peer message, local message, or not specified. For the last case, the direction is determined comparing the local sessionID and the message sessionID.

Returns

- resS_OK: The answer was properly updated.
- resFE_INVALID_STATE: InitializeSdpSession (see page 20) was not called or the answer was unexpected.
- resFE_INVALID_ARGUMENT: Problems were found with the SDP answer passed in parameter.

Description

This method is used to notify the B2B UA Connector that an answer was received and may need to be updated. The answer can come from the configured (local) SDP session or from another SDP session.

If the answer comes from the configured SDP session, no action is taken by the B2B UA Connector.

If the answer comes from another SDP session, then this method updates the answer to make it valid with regards to the configured SDP session.

This function modifies the answer, creates an "Update table", and changes the mode of operation to the "incremental update mode". The successive offer or answer updates will be performed in an incremental way using the "Update table".

The o= line is updated to reflect what the endpoint would expect as per the negotiated SDP session. If necessary, the version number in the "o=" line is also updated.

The v=, s=, i=, u=, e=, p=, b=, t=, r=, z= and k= lines are not modified.

See Also

UpdateOffer (see page 24), InitialUpdate, UpdateUsingTab.

9.1.1.4.11 - CSdpB2bUaConnector::UpdateOffer Method

Updates an offer received from the endpoint or from another entity.

C++

```
virtual mxt_result UpdateOffer(INOUT CSdpPacket* pOffer, IN EMessageToUpdateDirection eMessageDirection =
eMESSAGE_UNDEF_DIRECTION);
```

Parameters

Parameters	Description
`INOUT CSdpPacket* pOffer`	The offer received by the B2BUA.
`IN EMessageToUpdateDirection eMessageDirection = eMESSAGE_UNDEF_DIRECTION`	Indicates the message direction: peer message, local message, or not specified. For the last case, the direction is determined comparing the local sessionID and the message sessionID.

Returns

- resS_OK: The offer was properly updated.
- resFE_FAIL: The offer is not compatible and it can't be updated.
- resFE_INVALID_STATE: InitializeSdpSession (see page 20) was not called.
- resFE_INVALID_ARGUMENT: Problems were encountered with the SDP offer passed in parameter.

Description

This method is used to notify the B2B UA Connector that an offer has been received and may need to be updated. This offer can either belong to the configured SDP session, or to another SDP session.

If the offer comes from the configured SDP session, the B2B UA Connector will remember that there is an offer on the way and to expect an answer corresponding to this offer.

If the offer comes from another SDP session, then this method updates it in order to ensure that both SDP sessions are compatible and that both endpoints can exchange RTP directly.

There are two operation modes: "learning mode" and "incremental update mode". The current mode defines the update mechanism to be used.

The "learning mode" is the initial operation mode. In this mode, a compatibility check is made before trying the offer update. It uses the function InitialUpdate() to update the offer. This function modifies the offer, creates an "Update table", and changes the operation mode to the "incremental update mode". The successive offer or answer updates will be performed in an incremental way using the "Update table".

The o= line is updated to reflect what the endpoint would expect as per the negotiated SDP session. If necessary, the version number in the o= line is also updated.

The v=, s=, i=, u=, e=, p=, b=, t=, r=, z= and k= lines are not modified.

See Also

UpdateAnswer (see page 23), InitialUpdate and UpdateUsingTab.

9.1.1.5 - Enumerations

9.1.1.5.1 - CSdpB2bUaConnector::EMessageToUpdateDirection Enumeration

Message to update direction.

C++

```
enum EMessageToUpdateDirection {
  eMESSAGE_UNDEF_DIRECTION,
  eMESSAGE_FROM_LOCAL,
  eMESSAGE_FROM_PEER
};
```

Description

Message to update direction.

Members

Members	Description
`eMESSAGE_UNDEF_DIRECTION`	No specific direction.
`eMESSAGE_FROM_LOCAL`	The message to update was generated locally.
`eMESSAGE_FROM_PEER`	The message to update was generated by the peer.

9.1.1.5.2 - CSdpB2bUaConnector::EOfferAnswerDirection Enumeration

Offer answer direction

C++

```
enum EOfferAnswerDirection {
  eLOCAL_OFFER,
  eLOCAL_ANSWER
};
```

Description

Offer answer direction

Members

Members	Description
`eLOCAL_OFFER`	The offer was generated locally.
`eLOCAL_ANSWER`	The offer was generated by the peer.

9.1.1.6 - Friends

9.1.1.6.1 - friend struct SUpdateTabEntry Friend

This is needed to be able to compile with MSVC6 in which it is not possible for a struct declared in a class to access the enumeration values that are declared as private or protected in the same class.

C++

```
friend struct SUpdateTabEntry;
```

9.1.2 - CSdpCapabilitiesMgr Class

This class is used to find and manage the common capabilities between a UAC and a UAS.

Class Hierarchy

C++

```
class CSdpCapabilitiesMgr;
```

Description

The CSdpCapabilitiesMgr class is used to find and manage the common capabilities between a UAC and a UAS. This negotiation process is described in the "An Offer/Answer Model with SDP" Internet-Draft.

In this process, two UAs are involved in negotiating the media capabilities through SDP. One of the entities is the offerer. The offerer is the UA that is the first to offer a SDP packet to the other UA. The other entity is the answerer. The answerer receives the offer and provides an answer to the offerer. This answer has a matching media stream for each one in the offer, indicating if the stream is accepted and a list of codecs to use for each stream.

Initiating a Session

When the application is initiating a session, it can choose to be the offerer or the answerer. If it acts as the offerer, the application must include an SDP packet into the INVITE it is sending to the remote UA. The remote UA must then include the answer SDP packet into its 200 OK response.

If the application wants to act as the answerer, it must not include any SDP packet into the INVITE. It is up to the remote UA to include an offer in its 200 OK response. After receiving an offer into a 200 OK response, the application must include its answer into the following ACK request.

Remote UA Initiates a Session

A remote User-Agent initiating a session has the same choice as described in the previous section; it can choose to be the offerer or the answerer. In either case, when the local application wants to accept the incoming session, it MUST include an SDP packet into its 200 OK response. If the initiating UA did not include an SDP packet in the INVITE, then the local application acts as the offerer. If an SDP packet was included in the INVITE, then the application acts as the answerer.

Acting as the Offerer

When the application is acting as the offerer, it must include its full capability set into the SDP packet it sends.

Acting as the Answerer

The answerer must take the capabilities received from the offerer (the offer) and match those with its own local capabilities. The union from both of these capabilities yields the answer SDP packet. The main responsibility of the CSdpCapabilitiesMgr class is to do just that: to generate a proper answer according to the rules defined in SIP.

Session Modifications (re-INVITE)

After a session is established, it is possible to send a re-INVITE to modify a session and its associated capabilities. The same pattern as when initiating a session is used, but the offerer takes the accepted answer as the base of the new SDP packet to send.

Streams

A stream in the context of the CSdpCapabilitiesMgr corresponds to an "m=" line of a session description. A session description can contain more than one stream, as clearly defined in the SDP RFC 2327.

The SDP manager does the following to help strictly follow the SIP and SDP RFCs:

- It automatically aligns the media streams description.
- An unsupported stream automatically has its port set to zero.

- Only the codecs supported by both parties are part of the resulting negotiation.
- Rtpmap attributes are always added, helping for the support of dynamic payload types.

VAD Support for G.723, G.729 and G.729E

A stream sporting an rtpmap for G.723, G.729 or G.729E may also include an a=fmtp attribute for negotiation of the VAD annex. The CSdpCapabilitiesMgr can understand this fmtp attribute for both G.723 and G.729 and negotiate it following local configuration and received offer. By default this is turned off, see SetVadNegotiation (see page 60)() for more information.

fmtp attribute for telephone-event payload

A stream including an rtpmap for "telephone-event" may also include an a=fmtp attribute for negotiation of the supported events. When the fmtp attribute is absent (but the rtpmap is present), the events 0 through 15 are considered supported (as per RFC 2833). Otherwise, the CSdpCapabilitiesMgr can understand this fmtp attribute and send an answer that enables only events supported by both parties. The telephone-event support is turned off until a "telephone-event" rtpmap is added to a media line in the local config. When the rtpmap is in a media line of the local config, the local event support can be specified by adding a "telephone-event" fmtp attribute to the media line. To be able to use telephone event, you must also define MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE (see page 10) in PreSdpParserCfg.h.

fmtp attribute for redundancy ("red") payload

A stream including an rtpmap for "red" may also include an a=fmtp attribute for negotiation of the supported payload types in redundancy. When the fmtp attribute is absent (but the rtpmap is present), the payload types that are in the media line are considered supported (as per RFC 2198). Otherwise, the CSdpCapabilitiesMgr can understand this fmtp attribute and send an answer that contains only payload types supported in redundancy by both parties. The redundancy support is turned off until a "red" rtpmap is added to a media line in the local config. When the rtpmap is in a media line of the local config, the local redundancy payload type support can be specified by adding a "red" fmtp attribute. To be able to use redundancy event, you must also define MXD_SDP_ENABLE_REDUNDANCY_FMTP_ATTRIBUTE (see page 9) in PreSdpParserCfg.h.

How the media are matched when keymanagement is present

An offer is received with mikey set at the media level. The local capabilities manager first checks each media that it supports. It tries to find a media that has the same codec and the mikey attribute. If it does not find one, it rechecks every media but it uses the mikey attribute that has been set at the session level (if present) as if it was set in the media.

Note that when a mikey attribute is added to the local capabilities manager, it is not mandatory to use mikey for that media. If an offer is received that contains the same codec but without the mikey attribute, it matches the local media even if mikey is locally set on that media. The generated answer does not contain mikey though.

Location

SdpManagement/CSdpCapabilitiesMgr.h

Generating an Offer

In this example, the application is sending an offer. Keep in mind that an offer can be sent either in an INVITE request or in a 200 OK to an INVITE that did not contain any SDP in its payload.

```
CSdpCapabilitiesMgr localCapabilities;

// Adds a RTP audio stream with support for only one codec. The client
// should add all streams it is willing to support along with all the
// codecs for each stream. We also set the RTP published for this stream
unsigned int uStreamIdx;
localCapabilities.AddRtpAudioStream(unRtpPort, OUT unStreamIdx);

// The stream was added. Add the payloads that the application supports on
// this stream. In this case, we simply add PCMU support.
unsigned int uPayloadTypeIdx;
localCapabilities.AddPayloadType(unStreamIdx, CSdpParser::ePCMU,
                                              unPayloadTypeIdx);

// Create an SDP packet from the capabilities we have given to the SDP
// Caps manager
CSdpPacket resultPacket;
localCapabilities.CreateSdpPacket(szLocalAddr, szConnAddr, szVersion,
                                                           resultPacket);
```

```
// Put the SDP Packet into a blob. The SDP packet will be ready to be
// sent.
CBlob* pContent = MX_NEW(CBlob);
*pContent << resultPacket;
```

Receiving an Answer

After sending an offer, the application will eventually receive an answer. The application can verify that the answer was properly generated following the proper rules defined in the SIP specification. This is done by using VerifyAnswer (see page 61).

```
// we assume that the offer is contained in the SDP Caps manager
// instance "sdpOffer"

CSdpPacket receivedPacket;
const char* szReceivedSdpText =
    reinterpret_cast<const char*>(pReceivedSdpBlob->GetFirstIndexPtr());
mxt_result res;
receivedPacket.Parse(szReceivedText, res);

CSdpCapabilitiesMgr sdpAnswer;

if (MX_RIS_S(res))
{
    // If there was no parsing error, save the SDP packet into the new SDP
    // caps manager instance.
    sdpAnswer.CopyCapsFromPacket(receivedPacket);
}

// Check if the SDP in the answer is aligned with the offer.
if (sdpOffer.VerifyAnswer(remoteCapabilities))
{
    // The response received is aligned with the capabilities sent in the
    // INVITE so we can proceed with connection. We should now ACK the
    // response and cache the localCapabilities.
}
else
{
    // The remote party failed to align its capabilities to ours, adding
    // or removing media streams (m= lines). Hence, we should ACK and BYE
    // the call.
}
```

Generating an Answer

In this example, the application is sending an answer. An answer can be sent in a 200 OK response to an INVITE that contained an SDP payload or it can be generated in an ACK request when the initial INVITE did not contain any SDP.

```
CSdpCapabilitiesMgr localCapabilities;

// Adds a RTP audio stream with support for only one codec. The client
// should add all streams it is willing to support along with all the
// codecs for each stream.
unsigned int uStreamIdx;
localCapabilities.AddRtpAudioStream(unRtpPort, OUT unStreamIdx);

unsigned int uPayloadTypeIdx;
localCapabilities.AddPayloadType(unStreamIdx, CSdpParser::ePCMU,
    unPayloadTypeIdx);

// Parse the offer and keep it in an SDP caps manager instance.
const char* szSdpText =
    reinterpret_cast<const char*>(pReceivedContent->Begin());
CSdpPacket remoteSdpPacket;
mxt_result res;
remoteSdpPacket.Parse(szSdpText, res);

CSdpCapabilitiesMgr offerSdpCaps;

if (MX_RIS_S(res))
{
    // If there was no parsing error, save the SDP packet into the new SDP
    // caps manager instance.
    offerSdpCaps.CopyCapsFromPacket(remoteSdpPacket);
}

// This SDP caps instance will hold the answer.
CSdpCapabilitiesMgr answer;
```

```
    answer.GenerateAnswer(offerSdpCaps, localCapabilities);

    unsigned int uSupportedStreamIdx;
    if (answer.GetFirstSupportedStream(OUT unSupportedStreamIdx))
    {
        // There is compatible media stream so we can warn the user that a new
        // call arrived and we should cache the answer. This answer can be
        // re-used to generate a re-INVITE.
    }
    else
    {
        // Unable to generate a proper answer because there are no compatible
        // streams.
    }
```

Enabling G.729/G.723 VAD negotiation

In this example, the application wants to configure the local capabilities to negotiate VAD for G.729 and G.723. First it needs to enable the feature, then it adds the fmtp attribute to the selected media line.

```
    CSdpCapabilitiesMgr localCapabilities;

    // Adds a RTP audio stream with support for only one codec. The client
    // should add all streams it is willing to support along with all the
    // codecs for each stream.
    uint32_t unStreamIdx;
    localCapabilities.AddRtpAudioStream(uRtpPort, OUT uStreamIdx);

    // Add rtpmaps
    uint32_t uPayloadTypeIdx;
    localCapabilities.AddPayloadType(uStreamIdx, CSdpParser::eG729,
        uPayloadTypeIdx);
    localCapabilities.AddPayloadType(uStreamIdx, CSdpParser::eG723,
        uPayloadTypeIdx);
    localCapabilities.AddPayloadType(uStreamIdx, CSdpParser::eG729E,
        uPayloadTypeIdx);

    // Enable SdpCapsMgr to understand the fmtp attribute for VAD.
    CSdpCapabilitiesMgr::SetVadNegotiation(true);

    // Now that the feature is enabled, update the media line with explicit annex=yes
    localCapabilities.SetStreamVadSupport(CSdpParser::eG729, uStreamIndex, true);
    localCapabilities.SetStreamVadSupport(CSdpParser::eG723, uStreamIndex, true);

    // Now that the feature is enabled, update the media line with explicit annex=no
    localCapabilities.SetStreamVadSupport(CSdpParser::eG729E, uStreamIndex, false);
```

Finding out if G.729/G.723 VAD was negotiated during the SDP merge

In this example, the application wants to know if the merged capabilities include VAD support for G.729. First it needs to find the media line it wants to use, then verify if VAD is enabled for that stream.

```
    // ... assuming offer was received
    CSdpCapabilitiesMgr answer;
    answer.GenerateAnswer(offerSdpCaps, localCapabilities);

    uint32_t uSupportedStreamIdx;
    if (answer.GetFirstSupportedStream(OUT uSupportedStreamIdx))
    {
        // Assuming the first stream is an audio line that contains a G729 rtpmap
        if(answer.IsVadSupportedInStream(CSdpParser::eG729, uSupportedStreamIdx))
        {
          // Enable VAD for G.729 in application.
        }
    }
```

Enabling telephone-event fmtp negotiation

In this example, the application wants to configure the local capabilities to negotiate the telephone-event fmtp attribute for events 0 through 15, and 21. To be able to use telephone event, you must also define MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE (see page 10) in PreSdpParserCfg.h.

```
    CSdpCapabilitiesMgr localCapabilities;
    // Add rtpmap
    uint32_t uPayloadTypeIdx;
    localCapabilities.AddPayloadType(uStreamIdx, CSdpParser::eTELEPHONE_EVENT,
        uPayloadTypeIdx);

    // The local configuration for the first stream
    CSdpFmtpTelEvent localTelEventFmtp;

    // Enable events 0 through 15 we can use the eDTMF group to do this
    localTelEventFmtp.SetTelEventSupport(CSdpFmtpTelEvent::eDTMF, true);

    // Enable event 21 directly
    localTelEventFmtp.SetTelEventSupport(21, true);

    localCapabilities.ReplaceFmtpTelEvent(IN uStreamIdx,
                                          IN uPayloadTypeIdx,
                                          INOUT localTelEventFmtp);

    // Now SdpCapsManager will correctly negotiate telephone-events.
```

Finding out which telephone-events are supported in an offer or answer

In this example, the application wants to know if the merged capabilities (answer) support the telephone-event 15 in the first media line. To be able to use telephone event, you must also define MXD_SDP_ENABLE_TELEPHONE_EVENT_FMTP_ATTRIBUTE (see page 10) in PreSdpParserCfg.h.

```
    // ... assuming offer was received
    CSdpCapabilitiesMgr answer;
    answer.GenerateAnswer(offerSdpCaps, localCapabilities);

    // If there is no fmtp attribute, telephone-events are not supported.
    if (answer.GetFmtpTelEvent(0) != NULL)
    {
        if (answer.GetFmtpTelEvent(0)->IsTelephoneEventSupported(15) == true)
        {
            // Enable event 15 in application.
        }
    }
```

Enabling T.38 media

In this example, the application wants to send a T.38 media offer to the remote side. To be able to do this, you must first define MXD_SDP_ENABLE_T38_SUPPORT (see page 10) in PreSdpParserconfig.h.

```
      CSdpCapabilitiesMgr localCaps;
      unsigned int uStreamIndex = 0;

      CSdpLevelMedia t38Media;
      CSdpFieldMediaAnnouncement t38Announcement;
      CSdpFieldAttributeMaxBitRate maxBitRate;
      CSdpFieldAttributeVersion faxVersion;
      CSdpFieldAttributeT38FacsimileRateMgmnt faxRateMgmnt;
      CSdpFieldAttributeT38FacsimileMaxBuffer faxMaxBuffer;
      CSdpFieldAttributeMaxDatagram maxDatagram;
      CSdpFieldAttributeT38ErrorControl faxUdpErrorControl;

      //Set up the capacities of the T.38 stream.
      maxBitRate.SetMaxBitRate(14400);
      faxVersion.SetVersion(0);
      faxRateMgmnt.SetFacsimileRateMgmnt("transferredTCF");
      faxMaxBuffer.SetMaxBuffer(72);
      maxDatagram.SetMaxDatagram(316);
      faxUdpErrorControl.SetErrorControl("t38UDPRedundancy");

      // Configure our media type.
      t38Announcement.SetMediaTypeId(CSdpParser::eIMAGE);
      t38Announcement.SetTransportPort(5004);
      t38Announcement.SetTransportProtocolId(CSdpParser::eUDPTL);
      t38Announcement.AddMediaFormat("t38");
      t38Announcement.Validate();

      // Set the configured values in the media.
      t38Media.SetMediaAnnouncement(t38Announcement);
      t38Media.SetMaxBitRate(maxBitRate);
      t38Media.SetMaxDatagram(maxDatagram);
      t38Media.SetVersion(faxVersion);
      t38Media.SetT38ErrorControl(faxUdpErrorControl);
```

```
    t38Media.SetT38FacsimileRateMgmnt(faxRateMgmnt);
    t38Media.SetT38FacsimileMaxBuffer(faxMaxBuffer);
    t38Media.SetSession(localCaps.GetSdpSession());
    t38Media.Validate();

    localCaps.AddStream(t38Media, OUT uStreamIndex);
```

Constructors

Constructor	Description
◆ CSdpCapabilitiesMgr (see page 32)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpCapabilitiesMgr (see page 33)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 61)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ AddCryptoAttribute (see page 33)	Adds a supported crypto attribute to a stream.
◆ AddGroup (see page 33)	Adds a group to the session.
◆ AddKeyMgmtAttribute (see page 34)	Adds a key management attribute and parameter to the session level.
◆ AddMediaFormat (see page 34)	Adds a generic media format supported by the stream.
◆ AddPayloadType (see page 35)	Adds a supported RTP payload type to a stream.
◆ AddPhone (see page 35)	Adds optional SDP phone-fields into the session.
◆ AddRtpAudioStream (see page 36)	Configures the manager with an audio stream the application is willing to support.
◆ AddStream (see page 36)	Configures the manager with a stream the application is willing to support.
◆ AddVadFmtp (see page 37)	Adds a G.729 or G.723 VAD CSdpFieldAttributeFmtp (see page 95) instance into the specified stream. Deprecated since 1.7.7
◆ CopyCapsFromPacket (see page 38)	Transfers the capabilities represented in an SDP packet to the manager.
◆ CopyCapsToPacket (see page 38)	Transfers the capabilities of the manager to an SDP packet.
◆ CopyMikeyAttributes (see page 38)	Copies the MIKEY attributes from the target manager.
◆ CreateSdpPacket (see page 39)	Creates an SDP packet from the capabilities contained in the manager.
◆ DisableStream (see page 39)	Disables the use of a stream.
◆ EnableT38 (see page 40)	Configures the T.38 enabled state.
◆ FindRtpmap (see page 40)	Finds the rtpmap index for the given encoding name in the given stream.
◆ GenerateAnswer (see page 41)	Generates an answer SDP from an offer and the local capabilities.
◆ GetCryptoAttribute (see page 41)	Retrieves the payload found at a specific index of a stream.
◆ GetCryptoAttributes (see page 41)	Retrieves a vector containing all the supported crypto RTP payloads of a stream.
◆ GetFirstSupportedStream (see page 42)	Finds the first supported stream (non-zero port).
◆ GetFmtpRedundancy (see page 42)	Gets the fmtp attribute used for redundancy.
◆ GetFmtpTelEvent (see page 43)	Gets the fmtp attribute used for telephone-event.
◆ GetMaxAnswerRtpMaps (see page 43)	Returns current rtpmaps configuration.
◆ GetNbPayloadTypes (see page 43)	Retrieves the number of payloads a stream has.
◆ GetNbPhones (see page 44)	Retrieves the number of phone fields configured in the SDP session.
◆ GetNbStreams (see page 44)	Retrieves the number of streams configured in the SDP session.
◆ GetPayloadType (see page 44)	Retrieves the payload found at a specific index of a stream.
◆ GetPayloadTypes (see page 45)	Retrieves a vector containing all the supported RTP payloads of a stream.
◆ GetPhone (see page 45)	Retrieves a phone field.
◆ GetSdpSession (see page 45)	Gets the pointer to the level session within the Caps manager.

◆ GetSilenceSuppressionNegotiation (see page 46)	Fetches current setting for processing of silence suppression attribute for PCMU and PCMA.
◆ GetStream (see page 46)	Retrieves a stream.
◆ GetStreamAddr (see page 46)	Retrieves the IP address associated with a stream.
◆ GetStreamPort (see page 47)	Retrieves the port number associated with a specific stream.
◆ GetStreamPtimeMs (see page 47)	Returns the stream's ptime attribute. Value nSDP_CAPS_MGR_INVALID_PTIME means it is disabled.
◆ GetStreamTransportProtocol (see page 47)	Retrieves the type of transport used for a specific stream.
◆ GetStreamType (see page 48)	Retrieves the media type of a stream (audio, video, data, etc.).
◆ GetVadNegotiation (see page 48)	Fetches current setting for processing of fmtp attribute for G.729 annexb and G.723 annexa.
◆ IsSilenceSuppressionSupportedInStream (see page 48)	Checks if silence suppression is supported for the stream.
◆ IsStreamSupported (see page 49)	Verifies if a specific stream is supported (non-zero port).
◆ IsT38BooleanImplicitEncoding (see page 49)	Indicates if the T.38 boolean encoding is set to implicit.
◆ IsT38Enabled (see page 50)	Checks if T.38 is active. This should always be called on an answer generated locally or by the peer.
◆ IsVadSupportedInStream (see page 50)	Gets fmtp setting for G.729 annexb and G.723 annexa in specified stream. Deprecated since 1.7.7
◆ RemoveAllPayloadTypes (see page 51)	Removes all payload types from the specified stream.
◆ RemoveFmtpRedundancy (see page 52)	Removes the redundancy fmtp attribute from the selected stream.
◆ RemoveFmtpTelEvent (see page 52)	Removes the telephone-event fmtp attribute from the selected stream.
◆ RemovePayloadType (see page 52)	Removes the specified payload type from the specified stream.
◆ ReorderInLocalPayloadTypePriorityOrder (see page 53)	Reorders payload types in local capabilities order.
◆ ReplaceFmtpRedundancy (see page 53)	Replaces the fmtp attribute for redundancy in the selected stream.
◆ ReplaceFmtpTelEvent (see page 54)	Replaces the fmtp attribute for telephone-event in the selected stream.
◆ Reset (see page 55)	Clears all capabilities contained in the manager.
◆ ResetMikey (see page 55)	Resets the MIKEY data.
◆ SetMaxAnswerRtpMaps (see page 55)	Configures the maximum number of rtpmaps per media line to return in the answers.
◆ SetMicroLiteDefaultFamily (see page 56)	Sets the default destination address family
◆ SetMikey (see page 56)	Sets the IMikey interface used for this manager.
◆ SetMikeyKeys (see page 56)	Sets the keys to send with MIKEY.
◆ SetPeerCertificate (see page 56)	Sets the peer's certificate chain needed to handle some MIKEY messages.
◆ SetPeerIdentity (see page 57)	Sets the peer identity needed to handle a MIKEY message.
◆ SetSilenceSuppressionNegotiation (see page 57)	Enables processing of silence suppression attribute for PCMU and PCMA.
◆ SetStreamPort (see page 57)	Sets the port number associated with a stream.
◆ SetStreamPtimeMs (see page 58)	Sets the stream's ptime attribute. Set to nSDP_CAPS_MGR_INVALID_PTIME to disable use of ptime attribute.
◆ SetStreamSilenceSuppressionSupport (see page 58)	Sets silence suppression attribute for PCMU and PCMA.
◆ SetStreamVadAttribute (see page 58)	Sets fmtp attribute for G.729 annexb and G.723 annexb.
◆ SetStreamVadSupport (see page 59)	Sets fmtp attribute for G.729 annexb and G.723 annexb. Deprecated since 1.7.7
◆ SetT38BooleanEncoding (see page 59)	Sets the T.38 boolean encoding method. Updates the T.38 boolean encoding method of the T.38 streams.
◆ SetVadNegotiation (see page 60)	Enables processing of fmtp attribute for G.729 annexb and G.723 annexa.
◆ UseLocalPayloadTypePriorityInAnswer (see page 60)	Sets the priority to use for the payload types in the generated answers.
◆ UseLocalPayloadTypesInAnswer (see page 60)	Sets the usage of local payload type numbers in the generated answers.
◆ VerifyAnswer (see page 61)	Verifies that the received answer is coherent with the offer.

Legend

◆	Method

9.1.2.1 - Constructors

9.1.2.1.1 - CSdpCapabilitiesMgr

9.1.2.1.1.1 - CSdpCapabilitiesMgr::CSdpCapabilitiesMgr Constructor

Default constructor.

C++

```
CSdpCapabilitiesMgr();
```

Description

Default constructor.

9.1.2.1.1.2 - CSdpCapabilitiesMgr::CSdpCapabilitiesMgr Constructor

Copy Constructor.

C++

```
CSdpCapabilitiesMgr(IN const CSdpCapabilitiesMgr& rSource);
```

Parameters

Parameters	Description
`rFrom`	The object to copy.

Description

Copy constructor.

9.1.2.2 - Destructors

9.1.2.2.1 - CSdpCapabilitiesMgr::~CSdpCapabilitiesMgr Destructor

Destructor.

C++

```
virtual ~CSdpCapabilitiesMgr();
```

Description

Destructor.

9.1.2.3 - Methods

9.1.2.3.1 - CSdpCapabilitiesMgr::AddCryptoAttribute Method

Adds a supported crypto attribute to a stream.

C++

```
bool AddCryptoAttribute(IN unsigned int uStreamIndex, IN const CSdpFieldAttributeCrypto& rCryptoAtt);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	Zero-based index of the stream to which the crypto attribute should be added.
`IN const CSdpFieldAttributeCrypto& rCryptoAtt`	Crypto attribute to add.

Returns

True if a crypto attribute was added. False otherwise.

Description

Adds a supported crypto attribute at the end of the list of media attributes.

See Also

GetCryptoAttribute (see page 41), GetCryptoAttributes (see page 41)

9.1.2.3.2 - CSdpCapabilitiesMgr::AddGroup Method

Adds a group to the session.

C++

```
void AddGroup(IN const CSdpFieldAttributeGroup& rGroup);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeGroup& rGroup`	The group attribute field to add to the session.

Description

Adds a "a=group:" field to the session. More than one can be added to the same session.

9.1.2.3.3 - AddKeyMgmtAttribute

9.1.2.3.3.1 - CSdpCapabilitiesMgr::AddKeyMgmtAttribute Method

Adds a key management attribute and parameter to the session level.

C++

```
void AddKeyMgmtAttribute(IN CSdpFieldAttributeKeyMgmt& rKeyMgmt, IN const CSdpKeyManagementParameter& rKeyParam,
IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributeKeyMgmt& rKeyMgmt`	The key management attribute to add to the session.
`IN const CSdpKeyManagementParameter& rKeyParam`	The key management parameter to add to the session.
`IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH`	The role that the key management attribute will use.

Description

This method adds a key management attribute and its parameter to the session. It also creates a key management parameter for each underlying media stream if it does not have key management attribute. This adds the key management and the key management parameter to the end of the internal array. Note that any role previously set is overidden.

9.1.2.3.3.2 - CSdpCapabilitiesMgr::AddKeyMgmtAttribute Method

Adds a key management attribute and parameter to the media.

C++

```
void AddKeyMgmtAttribute(IN unsigned int uStreamIndex, IN CSdpFieldAttributeKeyMgmt& rKeyMgmt, IN const
CSdpKeyManagementParameter& rKeyParam, IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole =
CSdpFieldAttributeKeyMgmt::eBOTH);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to which the payload is to be added.
`IN CSdpFieldAttributeKeyMgmt& rKeyMgmt`	The key management attribute to add to the media stream.
`IN const CSdpKeyManagementParameter& rKeyParam`	The key management parameter to add to the media stream.
`IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH`	The role that the key management attribute will use.

Description

This method adds a key management attribute and its parameter to the specified stream. This adds the key management and the key management parameter to the end of the internal array. Note that any role previously set is overidden.

9.1.2.3.4 - CSdpCapabilitiesMgr::AddMediaFormat Method

Adds a generic media format supported by the stream.

C++

```
void AddMediaFormat(IN unsigned int uStreamIndex, IN const char* szFormat, OUT unsigned int& ruFormatIndex);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to which the format is to be added.

`IN const char* szFormat`	The format string to add.
`OUT unsigned int& ruFormatIndex`	The index of the new format in the specified stream.

Description

This method adds a supported 'format' to the specified stream. 'format' is defined in the SDP specification. This must not be used to add RTP payload types.

9.1.2.3.5 - AddPayloadType

9.1.2.3.5.1 - CSdpCapabilitiesMgr::AddPayloadType Method

Adds a supported RTP payload type to a stream.

C++

```
void AddPayloadType(IN unsigned int uStreamIndex, IN CSdpParser::ERtpCompressionAlgorithm eRtpAlgorithm, OUT
unsigned int& ruPayloadIndex, IN int nPayloadTypeNumber = -1);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to which the payload is to be added.
`IN CSdpParser::ERtpCompressionAlgorithm eRtpAlgorithm`	The supported codec algorithm to add. The payload number is automatically managed by the manager.
`OUT unsigned int& ruPayloadIndex`	The index of the new payload type in the specified stream. If no payload is added, ruPayloadIndex is set to an invalid value (0xFFFFFFFF).
`IN int nPayloadTypeNumber = -1`	Optionally sets a payload type number for the payload type. Default value is -1, meaning the default payload type number is used.

Description

This method adds a supported payload type to the specified stream. This has the effect of adding a payload number at the end of the "m=" line of the stream along with an "a=rtpmap:" line after the "m=" line.

9.1.2.3.5.2 - CSdpCapabilitiesMgr::AddPayloadType Method

Adds a supported RTP payload type to a stream.

C++

```
void AddPayloadType(IN unsigned int uStreamIndex, IN const CSdpFieldAttributeRtpmap& rRtpAlgorithm, OUT unsigned
int& ruPayloadIndex);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to which the payload is to be added.
`OUT unsigned int& ruPayloadIndex`	The index of the new payload type in the specified stream. If no payload is added, ruPayloadIndex is set to an invalid value (0xFFFFFFFF).
`rRtpMap`	An instance of the parser object describing the exact rtpmap to add. This specifies the payload string along with the payload number.

Description

This method adds a supported payload type to the specified stream. This has the effect of adding a payload number at the end of the "m=" line of the stream along with an "a=rtpmap:" line after the "m=" line. This version of AddPayload can be used when the application has to specify its own payload numbers.

9.1.2.3.6 - CSdpCapabilitiesMgr::AddPhone Method

Adds optional SDP phone-fields into the session.

C++

```
void AddPhone(IN const CSdpFieldPhone& rPhone);
```

Parameters

Parameters	Description
`IN const CSdpFieldPhone& rPhone`	A reference to the CSdpFieldPhone (see page 251) object that implements the phone number.

Description

AddPhone adds a "p=" line to the session description that is generated. It can be called multiple times to add more than one "p=" lines.

9.1.2.3.7 - CSdpCapabilitiesMgr::AddRtpAudioStream Method

Configures the manager with an audio stream the application is willing to support.

C++

```
void AddRtpAudioStream(IN int nPort, OUT unsigned int& ruStreamIndex);
```

Parameters

Parameters	Description
`IN int nPort`	The port number where the application is willing to receive this audio stream.
`OUT unsigned int& ruStreamIndex`	The index associated with the stream that was added. This index can later be used to reference this stream.

Description

Adds a stream of type CSdpParser::eAUDIO running over the CSdpParser::eRTPAVP protocol. This is just a quicker way to configure the manager for telephony applications that mainly only use the audio media over RTP.

9.1.2.3.8 - AddStream

9.1.2.3.8.1 - CSdpCapabilitiesMgr::AddStream Method

Configures the manager with a stream the application is willing to support.

C++

```
void AddStream(IN CSdpParser::EMediaType eMediaType, IN CSdpParser::ETransportProtocol eProtocol, IN int nPort,
OUT unsigned int& ruStreamIndex);
```

Parameters

Parameters	Description
`IN CSdpParser::EMediaType eMediaType`	The type of stream to add. Current valid values are: • CSdpParser::eAUDIO • CSdpParser::eVIDEO • CSdpParser::eAPPLICATION • CSdpParser::eDATA • CSdpParser::eCONTROL • CSdpParser::eIMAGE
`IN CSdpParser::ETransportProtocol eProtocol`	The transport protocol for the stream to add. Current valid values
`IN int nPort`	The port number where the application is willing to receive such a stream.
`OUT unsigned int& ruStreamIndex`	The index associated with the stream that was added. This index can later be used to reference this stream.
`are`	• CSdpParser::eRTPAVP • CSdpParser::eUDP • CSdpParser::eUDPTL • CSdpParser::eTCP

Description

AddStream is used to configure the CSdpCapabilitiesMgr (see page 25) instance with an additional stream that the application is willing to support. This has the effect of adding an "m=" line to the session description that will be generated.

9.1.2.3.8.2 - CSdpCapabilitiesMgr::AddStream Method

Configures the manager with a stream the application is willing to support.

C++

```
void AddStream(IN const CSdpLevelMedia& rMedia, OUT unsigned int& ruStreamIndex);
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rMedia`	A reference to the SDP object that implements the media level. This can be more flexible (but more complicated) than adding a stream with the first version of AddStream.
`OUT unsigned int& ruStreamIndex`	The index associated with the stream that was added. This index can later be used to reference this stream.

Description

AddStream is used to configure the CSdpCapabilitiesMgr (see page 25) instance with an additional stream that the application is willing to support. This has the effect of adding an "m=" line to the session description that is generated.

9.1.2.3.8.3 - CSdpCapabilitiesMgr::AddStream Method

Configures the manager with a stream the application is willing to support

C++

```
void AddStream(IN const char* szMediaType, IN const char* szProtocol, IN int nPort, OUT unsigned int&
ruStreamIndex);
```

Parameters

Parameters	Description
`IN const char* szMediaType`	The type of stream to add as a string.
`IN const char* szProtocol`	The transport protocol for the stream to add as a string.
`IN int nPort`	The port number where the application is willing to receive such a stream.
`OUT unsigned int& ruStreamIndex`	The index associated with the stream that was added. This index can later be used to reference this stream.

Description

AddStream is used to configure the CSdpCapabilitiesMgr (see page 25) instance with an additional stream that the application is willing to support. This has the effect of adding an "m=" line to the session description that will be generated.

9.1.2.3.9 - AddVadFmtp

9.1.2.3.9.1 - CSdpCapabilitiesMgr::AddVadFmtp Method Deprecated since 1.7.7

Adds a G.729 or G.723 VAD CSdpFieldAttributeFmtp (see page 95) instance into the specified stream. Deprecated since 1.7.7

C++

```
void AddVadFmtp(IN ECapsMgrPayloadType ePayloadType, IN bool bEnable, INOUT CSdpLevelMedia& rStream) const;
```

Parameters

Parameters	Description
`IN ECapsMgrPayloadType ePayloadType`	Payload number for which to add the VAD fmtp attribute. Only G.729 (18) and G.723 (4) are supported. Other payload types will produce unexpected results.
`IN bool bEnable`	Set to true to set an "enabled" VAD attribute, e.g. annexa=yes.
`INOUT CSdpLevelMedia& rStream`	Media line for which to set the fmtp attribute.

Description

Adds a VAD fmtp attribute for the specified payload. This means adding an a=fmtp line to the media line.

9.1.2.3.9.2 - CSdpCapabilitiesMgr::AddVadFmtp Method

Adds a G.729 or G.723 VAD CSdpFieldAttributeFmtp (see page 95) instance into the specified stream.

C++

```
void AddVadFmtp(IN const CString& rstrEncodingName, IN int nPayloadType, IN bool bEnable, INOUT CSdpLevelMedia&
rStream) const;
```

Parameters

Parameters	Description
`IN const CString& rstrEncodingName`	The encoding name for the payload type. Used when the payload type is in the dynamic range.
`IN int nPayloadType`	Payload number for which to add the VAD fmtp attribute. Payloads not supporting VAD through the annex fmtp attribute will produce unexpected results.
`IN bool bEnable`	Set to true to set an "enabled" VAD attribute, e.g. annexa=yes.
`INOUT CSdpLevelMedia& rStream`	Media line for which to set the fmtp attribute.

Description

Adds a VAD fmtp attribute for the specified payload. This means adding an a=fmtp line to the media line.

9.1.2.3.10 - CSdpCapabilitiesMgr::CopyCapsFromPacket Method

Transfers the capabilities represented in an SDP packet to the manager.

C++

```
void CopyCapsFromPacket(IN const CSdpPacket& rPacket);
```

Parameters

Parameters	Description
`IN const CSdpPacket& rPacket`	The SDP packet from which to get the capabilities.

Description

This method is used to capture the necessary information from the SDP packet given as a parameter. This means that the CSdpCapabilitiesMgr (see page 25) instance has the capabilities associated with the given SDP packet.

9.1.2.3.11 - CSdpCapabilitiesMgr::CopyCapsToPacket Method

Transfers the capabilities of the manager to an SDP packet.

C++

```
void CopyCapsToPacket(INOUT CSdpPacket& rPacket) const;
```

Parameters

Parameters	Description
`INOUT CSdpPacket& rPacket`	The SDP packet instance that is to receive the capabilities.

Description

This method is used to transfer the capabilities found in the CSdpCapabilitiesMgr (see page 25) instance to a CSdpPacket (see page 347) instance. This is usually used with a CSdpCapabilitiesMgr (see page 25) instance that either contains the local capabilities set (when initiating a call) or the common capabilities set (when accepting a call).

9.1.2.3.12 - CSdpCapabilitiesMgr::CopyMikeyAttributes Method

Copies the MIKEY attributes from the target manager.

C++

```
void CopyMikeyAttributes(IN const CSdpCapabilitiesMgr& rSource);
```

Parameters

Parameters	Description
`IN const CSdpCapabilitiesMgr& rSource`	The capabilities manager from which to copy the MIKEY data.

Description

Copies the IMikey, the peer identity, peer certificate and keys from the capabilites manager.

9.1.2.3.13 - CreateSdpPacket

9.1.2.3.13.1 - CSdpCapabilitiesMgr::CreateSdpPacket Method

Creates an SDP packet from the capabilities contained in the manager.

C++

```
void CreateSdpPacket(IN const char* szLocalIpAddress, IN const char* szConnectionIpAddress, IN const char*
szVersion, OUT CSdpPacket& rSdpPacket, OUT mxt_result* pres = NULL);
```

Parameters

Parameters	Description
`IN const char* szLocalIpAddress`	The address of the local host. This address is used to create the 'o=' line of the SDP packet.
`IN const char* szConnectionIpAddress`	The address where the remote UA should send its streams. This is used to create a 'c=' line at the session level. If this parameter is null, the address found in szLocalIpAddress is used instead.
`IN const char* szVersion`	The version of the SDP packet to be created. This is used to create the 'o=' line of the session description. This version can be incremented every time the session description changes.
`OUT CSdpPacket& rSdpPacket`	The instance where the SDP packet is to be created.
`OUT mxt_result* pres = NULL`	Optional parameter to hold the result.

Description

The method is used to fill a CSdpPacket (see page 347) instance with the capabilities defined in the SDP caps manager.

9.1.2.3.13.2 - CSdpCapabilitiesMgr::CreateSdpPacket Method

Creates an SDP packet from the capabilities contained in the manager.

C++

```
void CreateSdpPacket(IN const char* szLocalIpAddress, IN const char* szVersion, OUT CSdpPacket& rSdpPacket);
```

Parameters

Parameters	Description
`IN const char* szLocalIpAddress`	The address of the local host. This address is used to create the 'o=' line of the SDP packet.
`IN const char* szVersion`	The version of the SDP packet to be created. This is used to create the 'o=' line of the session description. This version can be incremented every time the session description changes.
`OUT CSdpPacket& rSdpPacket`	The instance where the SDP packet is to be created.

Description

This method was deprecated, please use CreateSdpPacket@IN const char*@IN const char*@IN const char*@OUT CSdpPacket (see page 347)&@OUT mxt_result* instead.

9.1.2.3.14 - DisableStream

9.1.2.3.14.1 - CSdpCapabilitiesMgr::DisableStream Method

Disables the use of a stream.

C++

```
void DisableStream(IN CSdpLevelMedia& rStream);
```

Parameters

Parameters	Description
`IN CSdpLevelMedia& rStream`	Reference to the media stream to disable.

Description

DisableStream is used to disable the usage of a stream. This method is used after agreeing on one or more streams for a session, when the application wants to disable a stream that was previously used. Before sending its re-INVITE, the application uses its CSdpCapabilitiesMgr (see page 25) instance holding the session capabilities and disables the stream before creating and sending the SDP packet.

Note that once a stream has been disabled in SIP, it should not be re-enabled or re-used. This has the effect of setting the port associated with this stream to zero.

9.1.2.3.14.2 - CSdpCapabilitiesMgr::DisableStream Method

Disables the use of a stream.

C++

```
void DisableStream(IN unsigned int uStreamIndex);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	Sets the index of the media stream to disable.

Description

DisableStream is used to disable the usage of a stream. This method is used after agreeing on one or more streams for a session, when the application wants to disable a stream that was previously used. Before sending its re-INVITE, the application uses its CSdpCapabilitiesMgr (see page 25) instance holding the session capabilities and disables the stream before creating and sending the SDP packet.

Note that once a stream has been disabled in SIP, it should not be re-enabled or re-used. This has the effect of setting the port associated with this stream to zero.

9.1.2.3.15 - CSdpCapabilitiesMgr::EnableT38 Method

Configures the T.38 enabled state.

C++

```
static void EnableT38(IN ECapsMgrT38MediaState eEnabledState);
```

Parameters

Parameters	Description
`IN ECapsMgrT38MediaState eEnabledState`	• eSDP_CAPS_MGR_T38_MEDIA_DISABLED: T.38 is disabled; • eSDP_CAPS_MGR_T38_MEDIA_ENABLED_ONLY_T38 : T.38 is negotiated alone only; • eSDP_CAPS_MGR_T38_MEDIA_ENABLED_WITH_OTHER_MEDIA: T.38 is negotiated alone or with other media.

Description

Configures whether or not the offers/answers created with all instances include T.38 capabilities.

When eEnabledState is eSDP_CAPS_MGR_T38_MEDIA_ENABLED_WITH_OTHER_MEDIA, all offers/answers include T.38 capabilities along with any other media streams configured by the user.

When eEnabledState is eSDP_CAPS_MGR_T38_MEDIA_ENABLED_ONLY_T38, all offers only include T.38 capabilities and T.38 media is deactivated if other media streams are present in an answer.

When eEnabledState is eSDP_CAPS_MGR_T38_MEDIA_DISABLED, all offers/answers do not include any T.38 capabilities and only include the other configured media streams.

If this method is not called, all the SDP capabilities managers assume that T.38 is not enabled.

9.1.2.3.16 - CSdpCapabilitiesMgr::FindRtpmap Method

Finds the rtpmap index for the given encoding name in the given stream.

C++

```
int FindRtpmap(IN const CSdpLevelMedia& rStream, IN const char* pszEncodingName) const;
```

9.1.2.3.17 - CSdpCapabilitiesMgr::GenerateAnswer Method

Generates an answer SDP from an offer and the local capabilities.

C++

```
void GenerateAnswer(IN const CSdpCapabilitiesMgr& rOffer, IN const CSdpCapabilitiesMgr& rLocalCaps, OUT
mxt_result* pres = NULL);
```

Parameters

Parameters	Description
IN const CSdpCapabilitiesMgr& rOffer	The CSdpCapabilitiesMgr (see page 25) instance containing the offer received. The common capabilities between rOffer and rLocalCaps use by default the same stream order and the same payload numbers as those defined in rOffer.
IN const CSdpCapabilitiesMgr& rLocalCaps	The CSdpCapabilitiesMgr (see page 25) instance that contains the local capabilities, when acting as the answerer.
OUT mxt_result* pres = NULL	Return the value of an internal protocol's processing. If NULL, no feedback from the underlying protocol is returned.

Description

This method is used to generate an answer containing the common capabilities retrieved from the common capabilities between the remote capabilities and the local capabilities. The instance where this method is called will receive the common capabilities.

By default this method uses rOffer as the capabilities that were received first. Thus, the inherited capabilities are based on these capabilities. The UseLocalPayloadTypesInAnswer (see page 60) and UseLocalPayloadTypePriorityInAnswer (see page 60) functions can be called to use local capabilities first.

If an ICE attribute is found in the offer, RTP and RTCP addresses of each media are validated upon ICE candidates. If no ICE candidates match the addresses, an ice-mismatch attribute is added in the answer in the corresponding media.

9.1.2.3.18 - CSdpCapabilitiesMgr::GetCryptoAttribute Method

Retrieves the payload found at a specific index of a stream.

C++

```
bool GetCryptoAttribute(IN unsigned int uStreamIndex, IN unsigned int uCryptoAttIndex, OUT
CSdpFieldAttributeCrypto& rCryptoAtt) const;
```

Parameters

Parameters	Description
IN unsigned int uStreamIndex	The stream index for which to retrieve the crypto attribute.
IN unsigned int uCryptoAttIndex	The index of the payload type to retrieve.
OUT CSdpFieldAttributeCrypto& rCryptoAtt	Output parameter where the Crypto information is stored.

Returns

True if a crypto attribute type was found. False otherwise.

Description

Retrieves a copy of the crypto attribute associated with a specific payload type.

9.1.2.3.19 - CSdpCapabilitiesMgr::GetCryptoAttributes Method

Retrieves a vector containing all the supported crypto RTP payloads of a stream.

C++

```
bool GetCryptoAttributes(IN unsigned int uStreamIndex, OUT CVector<CSdpFieldAttributeCrypto>& rvecCrypto) const;
```

Parameters

Parameters	Description
IN unsigned int uStreamIndex	The stream index for retrieving the payload types.
OUT CVector<CSdpFieldAttributeCrypto>& rvecCrypto	Output parameter that contains the retrieved crypto attributes.

Returns

True if all found crypto attribute types were copied into the vector. False otherwise.

Description

Retrieves all crypto attributes found within the specified stream.

9.1.2.3.20 - CSdpCapabilitiesMgr::GetFirstSupportedStream Method

Finds the first supported stream (non-zero port).

C++

```
bool GetFirstSupportedStream(OUT unsigned int& ruStreamIndex) const;
```

Parameters

Parameters	Description
`OUT unsigned int& ruStreamIndex`	The index of the first media stream that is supported.

Returns

True if a supported media stream was found. False otherwise.

Description

This method is used to retrieve the index of the first stream that is supported. A supported stream is one that does not have a port value of zero or is a send-only stream.

9.1.2.3.21 - GetFmtpRedundancy

9.1.2.3.21.1 - CSdpCapabilitiesMgr::GetFmtpRedundancy Method

Gets the fmtp attribute used for redundancy.

C++

```
CSdpFmtpRedundancy* GetFmtpRedundancy(IN uint32_t uStreamIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

The first redundancy fmtp attribute found in the selected stream.

NULL if no redundancy fmtp attribute was found in the selected stream.

Description

Gets the redundancy fmtp attribute in the selected stream.

9.1.2.3.21.2 - CSdpCapabilitiesMgr::GetFmtpRedundancy Method

Gets the fmtp attribute used for redundancy.

C++

```
const CSdpFmtpRedundancy* GetFmtpRedundancy(IN uint32_t uStreamIndex) const;
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

The first redundancy fmtp attribute found in the selected stream.

NULL if no redundancy fmtp attribute was found in the selected stream.

Description

Gets the redundancy fmtp attribute in the selected stream.

9.1.2.3.22 - GetFmtpTelEvent

9.1.2.3.22.1 - CSdpCapabilitiesMgr::GetFmtpTelEvent Method

Gets the fmtp attribute used for telephone-event.

C++

```
CSdpFmtpTelEvent* GetFmtpTelEvent(IN uint32_t uStreamIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

The first telephone-event fmtp attribute found in the selected stream.

NULL if no telephone-event fmtp attribute was found in the selected stream or if the stream is not found.

Description

Gets the telephone-event fmtp attribute in the selected stream.

9.1.2.3.22.2 - CSdpCapabilitiesMgr::GetFmtpTelEvent Method

Gets the fmtp attribute used for telephone-event.

C++

```
const CSdpFmtpTelEvent* GetFmtpTelEvent(IN uint32_t uStreamIndex) const;
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

The first telephone-event fmtp attribute found in the selected stream.

NULL if no telephone-event fmtp attribute was found in the selected stream.

Description

Gets the telephone-event fmtp attribute in the selected stream.

9.1.2.3.23 - CSdpCapabilitiesMgr::GetMaxAnswerRtpMaps Method

Returns current rtpmaps configuration.

C++

```
static unsigned int GetMaxAnswerRtpMaps();
```

Returns

Returns the maximum number of rtpmaps to set in an answer that is sent.

Description

Provides access to the max number of rtpmaps output in answers. A value of 0 means all supported rtpmaps are output from the offer.

Warning

This method is not thread safe.

9.1.2.3.24 - CSdpCapabilitiesMgr::GetNbPayloadTypes Method

Retrieves the number of payloads a stream has.

C++

```
unsigned int GetNbPayloadTypes(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to query.

Returns

The number of payloads associated with the specified stream.

Description

Retrieves the number of payloads associated with a specific stream.

9.1.2.3.25 - CSdpCapabilitiesMgr::GetNbPhones Method

Retrieves the number of phone fields configured in the SDP session.

C++

```
unsigned int GetNbPhones() const;
```

Returns

The number of phone fields for this session.

Description

Retrieves the number of phone fields currently configured for this session.

9.1.2.3.26 - CSdpCapabilitiesMgr::GetNbStreams Method

Retrieves the number of streams configured in the SDP session.

C++

```
unsigned int GetNbStreams() const;
```

Returns

The number of streams for this session.

Description

Retrieves the number of streams currently configured for this session. Note that all streams are included in the count, even disabled streams with a port of zero.

9.1.2.3.27 - CSdpCapabilitiesMgr::GetPayloadType Method

Retrieves the payload found at a specific index of a stream.

C++

```
void GetPayloadType(IN unsigned int uStreamIndex, IN unsigned int uPayloadIndex, OUT CSdpFieldAttributeRtpmap&
rRtpAlgo) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The stream index for which to retrieve the payload type.
`IN unsigned int uPayloadIndex`	The index of the payload type to retrieve.
`OUT CSdpFieldAttributeRtpmap& rRtpAlgo`	Output parameter where the RTP map information should be stored.

Returns

True if a supported payload type was found. False otherwise.

Description

Retrieves the CSdpFieldAttributeRtpmap (see page 187) associated with a specific payload type. This allows the application to determine what algorithm, along with it frequency, is associated with a specific payload. The payload is identified by its index within a specific stream.

9.1.2.3.28 - CSdpCapabilitiesMgr::GetPayloadTypes Method

Retrieves a vector containing all the supported RTP payloads of a stream.

C++

```
void GetPayloadTypes(IN unsigned int uStreamIndex, OUT CVector<CSdpFieldAttributeRtpmap>& rvecAlgo) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The stream index for retrieving the payload types.
`OUT CVector<CSdpFieldAttributeRtpmap>& rvecAlgo`	A vector given by the application and filled by the manager, describing the various supported algorithms for the specified stream.

Description

Retrieves a vector of algorithms associated with a specific stream. The algorithms are embedded into rtpmaps descriptors, allowing the simultaneous retrieval of the payload type and the algorithm description.

9.1.2.3.29 - CSdpCapabilitiesMgr::GetPhone Method

Retrieves a phone field.

C++

```
const CSdpFieldPhone& GetPhone(IN unsigned int uPhoneIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uPhoneIndex`	The index of the phone field to retrieve.

Returns

A reference to the CSdpFieldPhone (see page 251) instance.

Description

Retrieves the CSdpFieldPhone (see page 251) instance associated with a specific index.

9.1.2.3.30 - GetSdpSession

9.1.2.3.30.1 - CSdpCapabilitiesMgr::GetSdpSession Method

Gets the pointer to the level session within the Caps manager.

C++

```
CSdpLevelSession* GetSdpSession();
```

Returns

Pointer to the SDP session descriptor. It cannot be NULL.

Description

Provides access to the session descriptor data member.

9.1.2.3.30.2 - CSdpCapabilitiesMgr::GetSdpSession Method

Gets the pointer to the level session within the Caps manager.

C++

```
const CSdpLevelSession* GetSdpSession() const;
```

Returns

Pointer to the SDP session descriptor. It cannot be NULL.

Description

Provides access to the session descriptor data member.

9.1.2.3.31 - CSdpCapabilitiesMgr::GetSilenceSuppressionNegotiation Method

Fetches current setting for processing of silence suppression attribute for PCMU and PCMA.

C++

```
static bool GetSilenceSuppressionNegotiation();
```

Returns

Current activity status of the silence suppression negotiation feature.

Description

Fetches the current setting for processing of silence suppression attribute.

See Also

SetSilenceSuppressionNegotiation (see page 57)

9.1.2.3.32 - GetStream

9.1.2.3.32.1 - CSdpCapabilitiesMgr::GetStream Method

Retrieves a stream.

C++

```
CSdpLevelMedia& GetStream(IN unsigned int uStreamIndex);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to retrieve.

Returns

A reference to the CSdpLevelMedia (see page 288) instance associated with the stream.

Description

Retrieves the CSdpLevelMedia (see page 288) instance associated with a specific stream. By using this instance, the application can have a better control on the content and values of the media description.

9.1.2.3.32.2 - CSdpCapabilitiesMgr::GetStream Method

Retrieves a stream.

C++

```
const CSdpLevelMedia& GetStream(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to retrieve.

Returns

A const reference to the CSdpLevelMedia (see page 288) instance associated with the stream.

Description

Retrieves the CSdpLevelMedia (see page 288) instance associated with a specific stream. By using this instance, the application can have a better control on the content and values of the media description.

9.1.2.3.33 - CSdpCapabilitiesMgr::GetStreamAddr Method

Retrieves the IP address associated with a stream.

C++

```
bool GetStreamAddr(IN unsigned int uStreamIndex, OUT CString& rstrAddr) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the media stream from which to retrieve the address.
`OUT CString& rstrAddr`	The address for the specified media stream ('c' line in SDP).

Returns

True if the address was found, false otherwise, in which case the given index was invalid.

Description

Retrieves the address where a stream should be sent. This address can be either an IP address or a FQDN. Depending on when this method is used, this can be either the address where the local application wants to receive the stream or the address where the remote UA wants to receive the stream.

9.1.2.3.34 - CSdpCapabilitiesMgr::GetStreamPort Method

Retrieves the port number associated with a specific stream.

C++

```
int GetStreamPort(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to query.

Returns

The port number associated with the specified stream.

Description

Retrieves the port associated with a specific stream. Note that a port of zero means that the stream is disabled.

9.1.2.3.35 - CSdpCapabilitiesMgr::GetStreamPtimeMs Method

Returns the stream's ptime attribute. Value nSDP_CAPS_MGR_INVALID_PTIME means it is disabled.

C++

```
int32_t GetStreamPtimeMs(IN uint32_t uStreamIndex) const;
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	Index of the stream to verify.

Returns

Returns the value of the ptime attribute in the specified media line. If the media line is absent, or if ptime is absent or disabled, the return value is g_nINVALID_PTIME.

Description

Provides direct access to the ptime value within a stream.

9.1.2.3.36 - CSdpCapabilitiesMgr::GetStreamTransportProtocol Method

Retrieves the type of transport used for a specific stream.

C++

```
CSdpParser::ETransportProtocol GetStreamTransportProtocol(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to query.

Returns

The transport type for the specified stream.

Description

Queries the type of transport associated with a stream.

9.1.2.3.37 - CSdpCapabilitiesMgr::GetStreamType Method

Retrieves the media type of a stream (audio, video, data, etc.).

C++

```
CSdpParser::EMediaType GetStreamType(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to query.

Returns

The type of media associated with this stream.

Description

Queries the type of media associated with the specified stream.

9.1.2.3.38 - CSdpCapabilitiesMgr::GetVadNegotiation Method

Fetches current setting for processing of fmtp attribute for G.729 annexb and G.723 annexa.

C++

```
static bool GetVadNegotiation();
```

Returns

Current activity status of the VAD negotiation feature.

Description

Fetches current setting for processing of fmtp attribute for G.729 annexb and G.723 annexa.

See Also

SetVadNegotiation (see page 60)

9.1.2.3.39 - IsSilenceSuppressionSupportedInStream

9.1.2.3.39.1 - CSdpCapabilitiesMgr::IsSilenceSuppressionSupportedInStream Method

Checks if silence suppression is supported for the stream.

C++

```
bool IsSilenceSuppressionSupportedInStream(IN const CSdpLevelMedia& rStream) const;
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rStream`	Stream to validate for the silence suppression attribute.

Returns

True if silence suppression is currently supported in the stream, false otherwise.

Description

Gets the state of silence suppression for the media.

See Also

GetSilenceSuppressionNegotiation (see page 46), SetSilenceSuppressionNegotiation (see page 57)

9.1.2.3.39.2 - CSdpCapabilitiesMgr::IsSilenceSuppressionSupportedInStream Method

Checks if silence suppression is supported for the stream.

C++

```
bool IsSilenceSuppressionSupportedInStream(IN uint32_t uStreamIndex) const;
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	Stream index for which to validate the silence suppression attribute.

Returns

true if silence suppresion is supported in the stream, false otherwise.

Description

Verifies if silence supporesion is on for the stream at the specified index.

See Also

SetStreamSilenceSuppressionSupport (see page 58)

9.1.2.3.40 - IsStreamSupported

9.1.2.3.40.1 - CSdpCapabilitiesMgr::IsStreamSupported Method

Verifies if a specific stream is supported (non-zero port).

C++

```
bool IsStreamSupported(IN const CSdpLevelMedia& rStream) const;
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rStream`	The stream to verify.

Returns

True if the stream is supported (port is different than zero).

Description

This method queries the CSdpLevelMedia (see page 288) instance to know if the media it describes is supported. A supported stream has a non-zero port.

9.1.2.3.40.2 - CSdpCapabilitiesMgr::IsStreamSupported Method

Verifies if a specific stream is supported (non-zero port).

C++

```
bool IsStreamSupported(IN unsigned int uStreamIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream to query.

Returns

True if the stream found at unStreamIndes is supported.

Description

This method queries the manager to know if the stream at uStreamIndex is supported (port is different than 0).

9.1.2.3.41 - CSdpCapabilitiesMgr::IsT38BooleanImplicitEncoding Method

Indicates if the T.38 boolean encoding is set to implicit.

C++

```
bool IsT38BooleanImplicitEncoding();
```

Returns

True if the T.38 boolean encoding is set to implicit. False if the T.38 boolean encoding is set to explicit.

Description

Indicates if the T.38 boolean encoding is set to implicit.

9.1.2.3.42 - CSdpCapabilitiesMgr::IsT38Enabled Method

Checks if T.38 is active. This should always be called on an answer generated locally or by the peer.

C++

```
bool IsT38Enabled();
```

Returns

true if the answer contains an active T.38 media, false if no T.38 is active.

Description

This method parses the available medias in the answer. If any T.38 media are active in the offer, this returns true.

9.1.2.3.43 - IsVadSupportedInStream

9.1.2.3.43.1 - CSdpCapabilitiesMgr::IsVadSupportedInStream Method Deprecated since 1.7.7

Gets fmtp setting for G.729 annexb and G.723 annexa in specified stream. Deprecated since 1.7.7

C++

```
bool IsVadSupportedInStream(IN ECapsMgrPayloadType ePayloadType, IN const CSdpLevelMedia& rStream) const;
```

Parameters

Parameters	Description
`IN ECapsMgrPayloadType ePayloadType`	Payload number for which to check for the VAD fmtp attribute. Only G.729 (18) and G.723 (4) are supported. Other payload types will produce unexpected results.
`IN const CSdpLevelMedia& rStream`	Stream in which to check for VAD support.

Returns

Returns true if the VAD attribute is not found, or it is found and its value is "yes". RFC 3555 states that absence is to be treated as "yes".

Description

Checks if the specified stream supports VAD for the specified payload.

See Also

SetStreamVadSupport (see page 59)

9.1.2.3.43.2 - CSdpCapabilitiesMgr::IsVadSupportedInStream Method Deprecated since 1.7.7

Gets fmtp setting for G.729 annexb and G.723 annexa using specified stream index. Deprecated since 1.7.7

C++

```
bool IsVadSupportedInStream(IN ECapsMgrPayloadType ePayloadType, IN uint32_t uStreamIndex) const;
```

Parameters

Parameters	Description
`IN ECapsMgrPayloadType ePayloadType`	Payload number for which to chech for the VAD fmtp attribute. Only G.729 (18) and G.723 (4) are supported. Other payload types will produce unexpected results.
`IN uint32_t uStreamIndex`	Index of the stream in which to check for VAD support.

Returns

Returns true if the VAD attribute is not found, or it is found and its value is "yes". RFC 3555 states that absence is to be treated as "yes".

Description

Checks if the specified stream supports VAD for the specified payload.

9.1.2.3.43.3 - CSdpCapabilitiesMgr::IsVadSupportedInStream Method

Gets fmtp setting for G.729 annexb and G.723 annexa in specified stream.

C++

```
bool IsVadSupportedInStream(IN const CString& rstrEncodingName, IN int nPayloadType, IN const CSdpLevelMedia&
rStream) const;
```

Parameters

Parameters	Description
`IN const CString& rstrEncodingName`	The encoding name for the payload type. Used when the payload type is in the dynamic range.
`IN int nPayloadType`	Payload number for which to check for the VAD fmtp attribute. Payloads not supporting VAD through the annex fmtp attribute will produce unexpected results.
`IN const CSdpLevelMedia& rStream`	Stream in which to check for VAD support.

Returns

Returns true if the VAD attribute is not found, or it is found and its value is "yes". RFC 3555 states that absence is to be treated as "yes". This default behaviour can however be overridden by using the MXD_SDP_SILENCE_SUPPRESSION_INDICATION_ABSENCE_MEANS_DISABLED (see page 11) macro.

Description

Checks if the specified stream supports VAD for the specified payload.

See Also

SetStreamVadSupport (see page 59)

9.1.2.3.43.4 - CSdpCapabilitiesMgr::IsVadSupportedInStream Method

Gets fmtp setting for G.729 annexb and G.723 annexa using specified stream index.

C++

```
bool IsVadSupportedInStream(IN const CString& rstrEncodingName, IN int nPayloadType, IN uint32_t uStreamIndex)
const;
```

Parameters

Parameters	Description
`IN const CString& rstrEncodingName`	The encoding name for the payload type. Used when the payload type is in the dynamic range.
`IN int nPayloadType`	Payload number for which to check for the VAD fmtp attribute. Payloads not supporting VAD through annex fmtp attribute will produce unexpected results.
`IN uint32_t uStreamIndex`	Index of the stream in which to check for VAD support.

Returns

Returns true if the VAD attribute is not found, or it is found and its value is "yes". RFC 3555 states that absence is to be treated as "yes". This default behaviour can however be overridden by using the MXD_SDP_SILENCE_SUPPRESSION_INDICATION_ABSENCE_MEANS_DISABLED (see page 11) macro.

Description

Checks if the specified stream supports VAD for the specified payload.

9.1.2.3.44 - CSdpCapabilitiesMgr::RemoveAllPayloadTypes Method

Removes all payload types from the specified stream.

C++

```
bool RemoveAllPayloadTypes(IN uint32_t uStreamIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The stream index for which to remove all payload types.

Returns

false: The stream index was invalid. true: The payload types were removed from the stream.

Description

Removes all the payload types from the stream at index uStreamIndex.

9.1.2.3.45 - CSdpCapabilitiesMgr::RemoveFmtpRedundancy Method

Removes the redundancy fmtp attribute from the selected stream.

C++

```
bool RemoveFmtpRedundancy(IN uint32_t uStreamIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream from which to remove the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

false: The stream index was invalid.

true: The fmtp attribute was removed from the stream.

Description

Removes the redundancy fmtp attribute from the selected stream. The first redundancy fmtp attribute found in the stream is removed.

9.1.2.3.46 - CSdpCapabilitiesMgr::RemoveFmtpTelEvent Method

Removes the telephone-event fmtp attribute from the selected stream.

C++

```
bool RemoveFmtpTelEvent(IN uint32_t uStreamIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().

Returns

false: The stream index was invalid.

true; The fmtp attribute was removed from the stream.

Description

Removes the telephone-event fmtp attribute from the selected stream. The first telephone-event fmtp attribute found in the stream is removed.

9.1.2.3.47 - CSdpCapabilitiesMgr::RemovePayloadType Method

Removes the specified payload type from the specified stream.

C++

```
bool RemovePayloadType(IN uint32_t uStreamIndex, IN uint32_t uPayloadTypeIndex);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The stream index for which to remove the payload type.
`uPayloadIndex`	The index of the payload to remove in the specified stream.

Returns

false: The stream index or payload type index was invalid. true: The payload type was removed from the stream.

Description

Removes the payload type at index uPayloadIndex from the stream at index uStreamIndex.

9.1.2.3.48 - CSdpCapabilitiesMgr::ReorderInLocalPayloadTypePriorityOrder Method

Reorders payload types in local capabilities order.

C++

```
void ReorderInLocalPayloadTypePriorityOrder(IN unsigned int uResultStreamIdx, IN const CSdpCapabilitiesMgr&
rLocalCaps);
```

Parameters

Parameters	Description
`IN unsigned int uResultStreamIdx`	The index of the stream with the result to be reordered.
`IN const CSdpCapabilitiesMgr& rLocalCaps`	The CSdpCapabilitiesMgr (see page 25) instance that contains the local capabilities, when acting as the answerer.

Description

Reorder the response payloads to follow local priorities. Here are examples:

Config: default mode. Local : 0 (PCMU),8 (PCMA), 102 (red) Offer : 98 (red), 18 (PCMA), 10 (PCMU) Answer: 98 (red), 18 (PCMA), 10 (PCMU)

Config: use local payload priority. Local : 0 (PCMU),8 (PCMA), 102 (red) Offer : 98 (RED), 18 (PCMA), 10 (PCMU) Answer: 10 (PCMU), 18 (PCMA), 98 (RED) In the answer we have the offerer type (98-18-10) but in the local preference order(PCMU-PCMA-red).

9.1.2.3.49 - ReplaceFmtpRedundancy

9.1.2.3.49.1 - CSdpCapabilitiesMgr::ReplaceFmtpRedundancy Method

Replaces the fmtp attribute for redundancy in the selected stream.

C++

```
bool ReplaceFmtpRedundancy(IN uint32_t uStreamIndex, IN const CSdpFmtpRedundancy& rFmtpRed);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().
`IN const CSdpFmtpRedundancy& rFmtpRed`	The fmtp redundancy attribute that contains which events are supported for the selected stream. The attribute MUST be a valid redundancy fmtp attribute. It will be added to the stream. The parameter has the media format it will have in the packet so it must be the same as the one in the rtpmap attribute for redundancy.

Returns

false: The stream index was invalid.

false: The fmtp attribute was invalid.

true; The fmtp attribute was appended to the stream.

Description

Adds an fmtp attribute to the selected stream. The fmtp attribute must be valid when this method is called. The first redundancy fmtp attribute found in the stream is removed.

9.1.2.3.49.2 - CSdpCapabilitiesMgr::ReplaceFmtpRedundancy Method

Replaces the fmtp attribute for redundancy in the selected stream.

C++

```
bool ReplaceFmtpRedundancy(IN uint32_t uStreamIndex, IN uint32_t uPayloadIndex, INOUT CSdpFmtpRedundancy&
rFmtpRed);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().
`IN uint32_t uPayloadIndex`	The index of the redundancy payload type in the selected stream.
`INOUT CSdpFmtpRedundancy& rFmtpRed`	The fmtp redundancy attribute that contains which events are supported for the selected stream. The attribute MUST contain a valid redundancy fmtp attribute value. It will be added to the stream. The media format used will change for the payload type of the selected payload type.

Returns

false: The stream index was invalid.

false: The payload index was invalid.

false: The fmtp attribute was invalid.

true: The fmtp attribute was appended to the stream.

Description

Adds an fmtp attribute to the selected stream. The fmtp attribute must be valid when this method is called. The first redundancy fmtp attribute found in the stream is removed.

9.1.2.3.50 - ReplaceFmtpTelEvent

9.1.2.3.50.1 - CSdpCapabilitiesMgr::ReplaceFmtpTelEvent Method

Replaces the fmtp attribute for telephone-event in the selected stream.

C++

```
bool ReplaceFmtpTelEvent(IN uint32_t uStreamIndex, IN const CSdpFmtpTelEvent& rFmtpTelEvent);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream from which to remove the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().
`IN const CSdpFmtpTelEvent& rFmtpTelEvent`	The fmtp telephone-event attribute that contains which events are supported for the selected stream. The attribute MUST be a valid telephone-event fmtp attribute. It will be added to the stream. The parameter has the media format it will have in the packet so it must be the same as the one in the rtpmap attribute for telephone-event.

Returns

false: The stream index was invalid.

false: The fmtp attribute was invalid.

true; The fmtp attribute was appended to the stream.

Description

Adds an fmtp attribute to the selected stream. The fmtp attribute must be valid when this method is called. The first telephone-event fmtp attribute found in the stream is removed.

9.1.2.3.50.2 - CSdpCapabilitiesMgr::ReplaceFmtpTelEvent Method

Replaces the fmtp attribute for telephone-event in the selected stream.

C++

```
bool ReplaceFmtpTelEvent(IN uint32_t uStreamIndex, IN uint32_t uPayloadIndex, INOUT CSdpFmtpTelEvent&
rFmtpTelEvent);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	The index of the stream to which append the fmtp attribute. The index MUST be smaller than GetNbStreams (see page 44)().
`IN uint32_t uPayloadIndex`	The index of the telephone-event payload type in the selected stream.
`INOUT CSdpFmtpTelEvent& rFmtpTelEvent`	The fmtp telephone-event attribute that contains which events are supported for the selected stream. The attribute MUST contain a valid telephone-event fmtp attribute value. It will be added to the stream. The media format used will change for the payload type of the selected payload type.

Returns

false: The stream index was invalid.

false: The payload index was invalid.

false: The fmtp attribute was invalid.

true; The fmtp attribute was appended to the stream.

Description

Adds an fmtp attribute to the selected stream. The media format used in the fmtp attribute must be valid when this method is called. The first telephone-event fmtp attribute found in the stream is removed.

9.1.2.3.51 - CSdpCapabilitiesMgr::Reset Method

Clears all capabilities contained in the manager.

C++

```
void Reset();
```

Description

Clears any capabilities that the instance may hold.

9.1.2.3.52 - CSdpCapabilitiesMgr::ResetMikey Method

Resets the MIKEY data.

C++

```
void ResetMikey();
```

Description

Resets the IMikey, peer identity, peer certificate and IMikeyKeys from the capabilities manager.

9.1.2.3.53 - CSdpCapabilitiesMgr::SetMaxAnswerRtpMaps Method

Configures the maximum number of rtpmaps per media line to return in the answers.

C++

```
static void SetMaxAnswerRtpMaps(IN unsigned int uMaxRtpMaps);
```

Parameters

Parameters	Description
`IN unsigned int uMaxRtpMaps`	Maximum number of rtpmaps put in an answer that is sent.

Description

Use this method to control the number of rtpmaps that you want to put into an answer. By default, all supported rtpmaps in the offer are echoed in the answer. Set a value of 0 to echo all rtpmaps from the offer.

Warning

This method is not thread safe.

9.1.2.3.54 - CSdpCapabilitiesMgr::SetMicroLiteDefaultFamily Method

Sets the default destination address family

C++

```
void SetMicroLiteDefaultFamily(IN CSdpParser::EAddressType eAddressFamily);
```

Parameters

Parameters	Description
`IN CSdpParser::EAddressType eAddressFamily`	The default address family to use.

Description

Sets the MicroLite default family to use for default destination address.

9.1.2.3.55 - CSdpCapabilitiesMgr::SetMikey Method

Sets the IMikey interface used for this manager.

C++

```
mxt_result SetMikey(IN IMikey* pMikey);
```

Parameters

Parameters	Description
`IN IMikey* pMikey`	The IMikey used to get the data.

Returns

-resS_OK: operation successful -resFE_INVALID_STATE: NULL pointer

Description

This method sets the IMikey interface used by this instance of the capabilities manager to handle MIKEY key management attributes.

9.1.2.3.56 - CSdpCapabilitiesMgr::SetMikeyKeys Method

Sets the keys to send with MIKEY.

C++

```
mxt_result SetMikeyKeys(CVector<IMikeyKey*>* pVecpKeys);
```

Parameters

Parameters	Description
`CVector<IMikeyKey*>* pVecpKeys`	Vector containing all the MIKEY keys to set to this exchange.

Returns

resS_OK: Operation succeeded. resFE_INVALID_ARGUMENT: invalid pointer passer.

Description

Sets any keys the application wants to use for SRTP in MIKEY. If none are set, a default is generated for the first initiation message.

9.1.2.3.57 - CSdpCapabilitiesMgr::SetPeerCertificate Method

Sets the peer's certificate chain needed to handle some MIKEY messages.

C++

```
mxt_result SetPeerCertificate(IN const CCertificateChain* pCertificateChain);
```

Parameters

Parameters	Description
`IN const CCertificateChain* pCertificateChain`	The certificate chain for the peer. This parameter is only present if MXD_PKI_ENABLE_SUPPORT is present.

Returns

resS_OK: Operation succeeded. resFE_INVALID_ARGUMENT: NULL pointer.

Description

Sets the certificate chain related to the peer being contacted.

9.1.2.3.58 - CSdpCapabilitiesMgr::SetPeerIdentity Method

Sets the peer identity needed to handle a MIKEY message.

C++

```
mxt_result SetPeerIdentity(IN IMikeyIdentity* pPeerIdentity);
```

Parameters

Parameters	Description
`IN IMikeyIdentity* pPeerIdentity`	A default peer attribute that can be set. This will then be set to each crypto session bundle when a new MIKEY message arrives. If an identity is available through SIP or a certificate, it is this one that should be used in case one is not specified inside the MIKEY message. This identity should be the one found in the peer's certificate

Returns

resS_OK: Operation succeeded. resFE_INVALID_ARGUMENT: NULL pointer.

Description

Sets the identity related to the peer being contacted.

9.1.2.3.59 - CSdpCapabilitiesMgr::SetSilenceSuppressionNegotiation Method

Enables processing of silence suppression attribute for PCMU and PCMA.

C++

```
static void SetSilenceSuppressionNegotiation(IN bool bEnable);
```

Parameters

Parameters	Description
`IN bool bEnable`	Enables the negotiation of the silence suppression attribute.

Description

Enables the processing of the silence suppression attribute as described in RFC 3108. The silence suppression attribute is used to indicate silence suppression support for PCMU and PCMA. Capabilities are negotiated as follows:

Table of negotiation results:

Offer LocalCfg Answer ABSENT on off (ABSENT maps to "off") ABSENT off off

on on on on off off

off on off off off off

See Also

GetSilenceSuppressionNegotiation (see page 46)

9.1.2.3.60 - CSdpCapabilitiesMgr::SetStreamPort Method

Sets the port number associated with a stream.

C++

```
void SetStreamPort(IN unsigned int uStreamIndex, IN int nPort);
```

Parameters

Parameters	Description
`IN unsigned int uStreamIndex`	The index of the stream.
`IN int nPort`	The port associated with the specified stream.

Returns

The port number associated with the specified stream.

Description

Sets the port associated with a specific stream.

9.1.2.3.61 - CSdpCapabilitiesMgr::SetStreamPtimeMs Method

Sets the stream's ptime attribute. Set to nSDP_CAPS_MGR_INVALID_PTIME to disable use of ptime attribute.

C++

```
void SetStreamPtimeMs(IN uint32_t uStreamIndex, IN int32_t nPtime);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	Index of the stream for which to set ptime.
`IN int32_t nPtime`	Ptime value to set in sent offers and answers, in milliseconds.

Description

Sets the value of the ptime media-level attribute in the specified stream. The ptime attribute is not negotiated. It is to be used as extra information in offers and answers sent.

9.1.2.3.62 - CSdpCapabilitiesMgr::SetStreamSilenceSuppressionSupport Method

Sets silence suppression attribute for PCMU and PCMA.

C++

```
void SetStreamSilenceSuppressionSupport(IN uint32_t uStreamIndex, IN bool bEnable);
```

Parameters

Parameters	Description
`IN uint32_t uStreamIndex`	Index of the stream to set the silence suppression support.
`IN bool bEnable`	Enables silence suppression for the stream.

Description

Sets the value "yes"/"no" for the silence suppression attribute for the media.

See Also

IsSilenceSuppressionSupportedInStream (see page 48)

9.1.2.3.63 - CSdpCapabilitiesMgr::SetStreamVadAttribute Method

Sets fmtp attribute for G.729 annexb and G.723 annexb.

C++

```
void SetStreamVadAttribute(IN CSdpParser::ERtpCompressionAlgorithm eCompressionAlgorithm, IN uint32_t
uStreamIndex, IN bool bEnable);
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpCompressionAlgorithm eCompressionAlgorithm`	Payload type identifier for which to set the VAD fmtp attribute. Only G.723 and G.729 variants are supported. Other payload identifiers will produce unexpected results.
`IN uint32_t uStreamIndex`	Index of the stream for which to enable VAD support. The method does nothing if the specified index is invalid.
`IN bool bEnable`	Set to true to set an "enabled" VAD attribute, e.g. annexa=yes.

Description

Sets the value "yes"/"no" for the VAD fmtp attribute for the specified payload type. This method does not check if the associated rtpmap is set or not, it must be added separately using the AddPayloadType (see page 35)() method.

See Also

IsVadSupportedInStream (see page 50)

9.1.2.3.64 - CSdpCapabilitiesMgr::SetStreamVadSupport Method Deprecated since 1.7.7

Sets fmtp attribute for G.729 annexb and G.723 annexb. Deprecated since 1.7.7

C++

```
void SetStreamVadSupport(IN ECapsMgrPayloadType ePayloadType, IN uint32_t uStreamIndex, IN bool bEnable);
```

Parameters

Parameters	Description
`IN ECapsMgrPayloadType ePayloadType`	Payload number for which to set the VAD fmtp attribute. Only G.729 (18) and G.723 (4) are supported. Other payload types will produce unexpected results.
`IN uint32_t uStreamIndex`	Index of the stream for which to enable VAD support. The method does nothing if the specified index is invalid.
`IN bool bEnable`	Set to true to set an "enabled" VAD attribute, e.g. annexa=yes.

Description

Sets the value "yes"/"no" for the VAD fmtp attribute for the specified payload type. This method does not check if the associated rtpmap is set or not, it must be added separately using the AddPayloadType (see page 35)() method.

See Also

IsVadSupportedInStream (see page 50)

9.1.2.3.65 - CSdpCapabilitiesMgr::SetT38BooleanEncoding Method

Sets the T.38 boolean encoding method. Updates the T.38 boolean encoding method of the T.38 streams.

C++

```
void SetT38BooleanEncoding(bool bT38BooleanImplicitEncoding);
```

Parameters

Parameters	Description
`bool bT38BooleanImplicitEncoding`	True indicates the T.38 boolean implicit encoding. False indicates the T.38 boolean explicit encoding.

Description

Sets the T.38 boolean encoding method. Updates the T.38 boolean encoding method of the T.38 streams.

The implicit method uses the presence or the absence of the boolean attribute to indicate true or false. For example, the presence of the T38FaxTranscodingMMR attribute indicates that the MMR transcoding is supported. If the T38FaxTranscodingMMR attribute is absent, it indicates the MMR transcoding is not supported.

The explicit method uses the name of the attribute and appends the ":1" or ":0" strings to indicate true or false. For example, "T38FaxTranscodingMMR:1" indicates that the MMR transcoding is supported. "T38FaxTranscodingMMR:0" indicates that the MMR transcoding is not supported.

The following table shows the different possibilities. The T38FaxFillBitRemoval attribute is used in the examples but this table also applies to the T38FaxTranscodingMMR and T38FaxTranscodingJBIG attributes.

Actual encoding:	New encoding:	Resulting encoding:
implicit false (no T38FaxFillBitRemoval attribute)	implicit	implicit false (no T38FaxFillBitRemoval attribute)
implicit false (no T38FaxFillBitRemoval attribute)	explicit	explicit false (a=T38FaxFillBitRemoval:0)
implicit true (a=T38FaxFillBitRemoval)	implicit	implicit true (a=T38FaxFillBitRemoval)
implicit true (a=T38FaxFillBitRemoval)	explicit	explicit true (a=T38FaxFillBitRemoval:1)
explicit false (a=T38FaxFillBitRemoval:0)	implicit	implicit false (no T38FaxFillBitRemoval attribute)
explicit false (a=T38FaxFillBitRemoval:0)	explicit	explicit false (a=T38FaxFillBitRemoval:0)
explicit true (a=T38FaxFillBitRemoval:1)	implicit	implicit true (a=T38FaxFillBitRemoval)
explicit true (a=T38FaxFillBitRemoval:1)	explicit	explicit true (a=T38FaxFillBitRemoval:1)

9.1.2.3.66 - CSdpCapabilitiesMgr::SetVadNegotiation Method

Enables processing of fmtp attribute for G.729 annexb and G.723 annexa.

C++

```
static void SetVadNegotiation(IN bool bEnable);
```

Parameters

Parameters	Description
`IN bool bEnable`	Activation status. True means enabled.

Description

Enables processing of fmtp attribute for G.729 annexb and G.723 annexa. When disabled, annexa for G.723 and annexb for G.729 are completely ignored in incoming offers, and not included in answers. If enabled, these fmtps are negotiated properly.

Once you have enabled the feature, you must then add the fmtp attribute to the media lines by using the SetStreamVadSupport (see page 59)() method.

RFC 3555 section 4.1.9 specifies the "annexb" parameter and its values for G.729. RFC3555 section 3 states that "Any payload-format-specific parameters go in the SDP "a=fmtp" attribute". RFC 3555 section 4.1.3 specifies use of the "annexa" parameter for G.723.

For both codecs, here is the Table of negotiation results:

Offer LocalCfg Answer ABSENT yes yes (RFC 3555 4.1.9: ABSENT maps to "yes") ABSENT no no

yes yes yes yes no no

no yes no no no no

See Also

GetVadNegotiation (see page 48), SetStreamVadSupport (see page 59)

9.1.2.3.67 - CSdpCapabilitiesMgr::UseLocalPayloadTypePriorityInAnswer Method

Sets the priority to use for the payload types in the generated answers.

C++

```
static void UseLocalPayloadTypePriorityInAnswer(IN bool bUseLocalPayloadTypePriority);
```

Parameters

Parameters	Description
`IN bool bUseLocalPayloadTypePriority`	True to use local priority order for the payload types in the generated answers. False to use the offer priority order for the payload types in the generated answers.

Description

This function sets priority to use for the payload types in the generated answers. The default value is to use the priority order of the offer, as the RFC 3264 states in the section 6.1: "Although the answerer MAY list the formats in their desired order of preference, it is RECOMMENDED that unless there is a specific reason, the answerer list formats in the same relative order they were present in the offer.".

Warning

This method is NOT thread safe.

9.1.2.3.68 - CSdpCapabilitiesMgr::UseLocalPayloadTypesInAnswer Method

Sets the usage of local payload type numbers in the generated answers.

C++

```
static void UseLocalPayloadTypesInAnswer(IN bool bUseLocalPayloadTypes);
```

Parameters

Parameters	Description
`IN bool bUseLocalPayloadTypes`	true to use local payload type numbers in the generated answers. false to use the offer payload type numbers in the generated answers. This is the default value.

Description

Set from where the payload type numbers must come.

Note that RFC 3264 states that "if a particular codec was referenced with a specific payload type number in the offer, that same payload type number SHOULD be used for that codec in the answer" (RFC 3264, section 6.1, 5th paragraph).

Warning

This method is NOT thread safe.

9.1.2.3.69 - CSdpCapabilitiesMgr::VerifyAnswer Method

Verifies that the received answer is coherent with the offer.

C++

```
bool VerifyAnswer(IN CSdpCapabilitiesMgr& rAnswer, OUT mxt_result* pres = NULL);
```

Parameters

Parameters	Description
`IN CSdpCapabilitiesMgr& rAnswer`	The CSdpCapabilitiesMgr (see page 25) that contains the SDP answer.
`OUT mxt_result* pres = NULL`	Optional parameter to hold the result:
`resS_OK`	The answer is accepted.
`resFE_FAIL`	The answer is refused.
`resFE_KEY_MANAGEMENT_NOT_SUPPORTED_BY_PEER`	The MIKEY key management is not supported by the peer. However, the streams in the answer are valid.
`resFE_KEY_MANAGEMENT_ERROR`	There was an error during the processing of the key management attribute. However, the streams in the answer are valid. Any result returned by ParseMikeyKeyMgmt.

Returns

True if the answer was properly generated from the offer contained in this instance, false otherwise.

Description

Verifies that the answer given in parameter was properly generated from the offer described in this instance of CSdpCapabilitiesMgr (see page 25). The number of streams and their order are verified for correctness.

If MXD_SDP_ENABLE_ICE_LITE_WITHOUT_CC_NEGOTIATION (see page 7) is defined, the method returns resFE_FAIL if the default destination does not have a corresponding ICE candidate.

9.1.2.4 - Operators

9.1.2.4.1 - CSdpCapabilitiesMgr::= Operator

Assignment operator.

C++

```
CSdpCapabilitiesMgr& operator =(IN const CSdpCapabilitiesMgr& rSource);
```

Parameters

Parameters	Description
`rFrom`	The object to copy.

Returns

A reference to this instance.

Description

Assignment operator.

9.1.2.5 - Friends

9.1.2.5.1 - friend class CSdpParserInitializer Friend

Friend class used by CSdpCapabilitiesMgr (see page 25)

C++

```
friend class CSdpParserInitializer;
```

9.2 - SDP SAFE Management Structs & Enums

Enumerations

Enumeration	Description
EPkgSdpMgmtFailErrorCodeId (see page 62)	Failure codes specific to the SdpMgmt package.

9.2.1 - EPkgSdpMgmtFailErrorCodeId Enumeration

Failure codes specific to the SdpMgmt package.

C++

```
enum EPkgSdpMgmtFailErrorCodeId {
  resFE_KEY_MANAGEMENT_NOT_SUPPORTED_BY_PEER = MX_RGET_PKG_BASE_FAIL_ERROR_CODE_ID( eMX_PKG_SDP ),
  resFE_MULTIPLE_KEY_MANAGEMENT_IN_MEDIA_ANSWER,
  resFE_KEY_MANAGEMENT_ERROR,
  resFE_ICE_NO_CANDIDATE_MATCH
};
```

Description

Failure codes specific to the SdpMgmt package.

Members

Members	Description
`resFE_KEY_MANAGEMENT_NOT_SUPPORTED_BY_PEER = MX_RGET_PKG_BASE_FAIL_ERROR_CODE_ID( eMX_PKG_SDP )`	Key management is not supported by the peer.
`resFE_MULTIPLE_KEY_MANAGEMENT_IN_MEDIA_ANSWER`	There is more than one key management in the media answer.
`resFE_KEY_MANAGEMENT_ERROR`	There is a key management error.
`resFE_ICE_NO_CANDIDATE_MATCH`	Cannot generate an answer since there is no ICE candidate that match the offer.

10 - SDP Parser APIs

This section documents the application level APIs that are available from the Parser of the SDP SAFE package, which is located under the SdpParser directory.

The SdpParser package offers many important objects which are used throughout SDP SAFE stack in order to parse, serialize, create and modify SDP payload easily.

10.1 - SDP Parser Classes

Classes

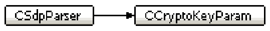
Class	Description
CCryptoKeyParam (see page 64)	This class implements key parameters.
CCryptoKeyParamList (see page 70)	This class implements a list of key parameters.
CCryptoSessionParam (see page 75)	This class implements session parameters.
CCryptoSessionParamList (see page 80)	This class implements a list of Session parameters.
CSdpFieldAttributeCrypto (see page 84)	This class implements an abstraction of an attribute-crypto.
CSdpFieldAttributeFillBitRemoval (see page 90)	This class implements an abstraction of an attribute-fill-bit-removal.
CSdpFieldAttributeFmtp (see page 95)	This class implements an abstraction of an attribute-fmtp.
CSdpFieldAttributeGroup (see page 101)	This class implements an abstraction of the group attribute.
CSdpFieldAttributeIceCandidate (see page 106)	Implements the ice-candidate attribute.
CSdpFieldAttributeIceOptions (see page 114)	Implements the ice-options attribute.
CSdpFieldAttributeIcePwd (see page 118)	Implements the ice-pwd attribute.
CSdpFieldAttributeIceRemoteCandidates (see page 121)	Implements the ICE remote-candidates attribute.
CSdpFieldAttributeIceSingleTokenBase (see page 128)	Base class for single token ICE attribute.
CSdpFieldAttributeIceUserFrag (see page 131)	Implements the ice-ufrag attribute.
CSdpFieldAttributeKeyMgmt (see page 134)	This class implements an abstraction of a key-mgmt-attribute.
CSdpFieldAttributeKeyMgmtMikey (see page 140)	This class implements an abstraction of a MIKEY key-mgmt-attribute.
CSdpFieldAttributeMaxBitRate (see page 149)	This class implements an abstraction of an attribute-max-bit-rate.
CSdpFieldAttributeMaxDatagram (see page 153)	This class implements abstraction of an attribute-max-datagram.
CSdpFieldAttributeMid (see page 157)	This class implements an abstraction of the mid attribute.
CSdpFieldAttributeOther (see page 160)	This class implements an abstraction of an attribute-other.
CSdpFieldAttributePreCond (see page 165)	This class implements an abstraction of the precondition field attribute.
CSdpFieldAttributePreCondConf (see page 169)	This class implements a CONF precondition field attribute.
CSdpFieldAttributePreCondCurr (see page 172)	This class implements a CURR precondition field attribute.
CSdpFieldAttributePreCondDes (see page 174)	This class implements a DES precondition field attribute.
CSdpFieldAttributePtime (see page 177)	This class implements an abstraction of an attribute-ptime.
CSdpFieldAttributeRtcp (see page 181)	This class implements an abstraction of the rtcp SDP attribute.
CSdpFieldAttributeRtpmap (see page 187)	This class implements an abstraction of an attribute-rtpmap.
CSdpFieldAttributeSilenceSupp (see page 194)	This class implements the parsing and serializing of the silence suppression attribute.
CSdpFieldAttributeT38ErrorControl (see page 198)	This class implements an abstraction of an attribute-t38-error-control.
CSdpFieldAttributeT38FacsimileMaxBuffer (see page 202)	This class implements an abstraction of an attribute-t38-facsimile-max-buffer.
CSdpFieldAttributeT38FacsimileRateMgmnt (see page 206)	This class implements an abstraction of an attribute-t38-Facsimile-rate-mgmnt.
CSdpFieldAttributeTranscoding (see page 210)	This class implements an abstraction of an attribute-transcoding.
CSdpFieldAttributeTranscodingJBIG (see page 213)	This class implements an abstraction of an attribute-transcoding-jbig.
CSdpFieldAttributeTranscodingMMR (see page 218)	This class implements an abstraction of a attribute-transcoding-mmr.
CSdpFieldAttributeVersion (see page 224)	This class implements an abstraction of a attribute-version.
CSdpFieldConnectionData (see page 228)	This class implements an abstraction of a connection-field.
CSdpFieldMediaAnnouncement (see page 235)	This class implements an abstraction of a media-field.
CSdpFieldOrigin (see page 243)	This class implements an abstraction of an origin-field.
CSdpFieldPhone (see page 251)	This class implements an abstraction of a session-name-field.
CSdpFieldProtocolVersion (see page 255)	This class implements an abstraction of a proto-version field.
CSdpFieldSessionName (see page 259)	This class implements an abstraction of a session-name-field.
CSdpFieldTime (see page 263)	This class implements an abstraction of the time-fields.
CSdpFmtpRedundancy (see page 269)	This class implements an abstraction of a redundancy fmtp attribute.
CSdpFmtpTelEvent (see page 274)	This class implements an abstraction of a telephone-event fmtp attribute.
CSdpKeyManagementParameter (see page 280)	This class implements the base class for handling parameters related to key management attribute.
CSdpKeyManagementParameterMikey (see page 283)	This class is a container object used to store the media level parameters used to generate a MIKEY message.

CSdpLevelMedia (see page 288)	This class implements an abstraction of a media-descriptions part.
CSdpLevelSession (see page 326)	This class implements an abstraction of the level-session.
CSdpPacket (see page 347)	This class implements an abstraction of a SDP packet.
CSdpParser (see page 351)	This class implements the abstract base class for all the other SDP Parser classes.

10.1.1 - CCryptoKeyParam Class

This class implements key parameters.

Class Hierarchy



C++

```
class CCryptoKeyParam : public CSdpParser;
```

Description

This class implements key parameters, separated by character specified in the draft-ietf-mmusic-sdescriptions-09.

```
   draft-ietf-mmusic-sdescriptions-09 ABNF:

       key-param       = key-method ":" key-info
       key-method      = srtp-key-method
       srtp-key-method = "inline"
       key-info        = srtp-key-info
       srtp-key-info   = key-salt ["|" lifetime] ["|" mki]

       key-salt        = 1*(base64)   ; binary key and salt values
                                      ; concatenated together, and then
                                      ; base64 encoded [section 6.8 of
                                      ; RFC2046]

       lifetime      = ["2^"] 1*(DIGIT)   ; see section 5.1 for "2^"
       mki           = mki-value ":" mki-length
       mki-value     = 1*DIGIT
       mki-length    = 1*3DIGIT   ; range 1..128.

       base64        = ALPHA / DIGIT / "+" / "/" / "="
```

Location

SdpParser/CCryptoKeyParam.h

Constructors

Constructor	Description
◆ CCryptoKeyParam (see page 65)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CCryptoKeyParam (see page 66)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 70)	Assignment operator.
◆ == (see page 70)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetKey (see page 66)	Gets the crypto key.
◆ GetKeyMethod (see page 66)	Gets the key method.
◆ GetLifeTime (see page 66)	Gets the lifetime.
◆ GetMkiLength (see page 67)	Gets mki length.
◆ GetMkiValue (see page 67)	Gets the mki value.
◆ Parse (see page 67)	Parses the parameters list beginning at rpszStartPosition.
◆ Reset (see page 68)	Resets this object.
◆ Serialize (see page 68)	Inserts cSeparator after each parameter, except for last parameter.
◆ SetKey (see page 68)	Sets the crypto key.
◆ SetKeyMethod (see page 68)	Sets the key method.
◆ SetLifeTime (see page 69)	Sets the lifetime.
◆ SetMki (see page 69)	Sets the mki value of the key and its length.
◆ SetMkiLength (see page 69)	Sets the mki length.
◆ SetMkiValue (see page 69)	Sets the mki value.
◆ Validate (see page 70)	Return true if data members are valid

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.1.1 - Constructors

10.1.1.1.1 - CCryptoKeyParam

10.1.1.1.1.1 - CCryptoKeyParam::CCryptoKeyParam Constructor

Default constructor.

C++

```
CCryptoKeyParam();
```

Description

Default constructor.

10.1.1.1.1.2 - CCryptoKeyParam::CCryptoKeyParam Constructor

Copy constructor

C++

```
CCryptoKeyParam(IN const CCryptoKeyParam& rCryptoKeyParam);
```

Parameters

Parameters	Description
rSrc	List to copy.

Description

Copy constructor.

10.1.1.2 - Destructors

10.1.1.2.1 - CCryptoKeyParam::~CCryptoKeyParam Destructor

Destructor.

C++

```
virtual ~CCryptoKeyParam();
```

Description

Destructor.

10.1.1.3 - Methods

10.1.1.3.1 - CCryptoKeyParam::GetKey Method

Gets the crypto key.

C++

```
const char* GetKey() const;
```

Returns

A char* with the key in it.

Description

Returns the key and salt key in base64 format.

10.1.1.3.2 - CCryptoKeyParam::GetKeyMethod Method

Gets the key method.

C++

```
const char* GetKeyMethod() const;
```

Returns

NULL-terminated string of the key method.

Description

Returns the key method.

10.1.1.3.3 - CCryptoKeyParam::GetLifeTime Method

Gets the lifetime.

C++

```
const uint64_t GetLifeTime() const;
```

Returns

The life time of the key.

Description

Returns the life time of the key.

10.1.1.3.4 - CCryptoKeyParam::GetMkiLength Method

Gets mki length.

C++

```
int GetMkiLength() const;
```

Returns

The Mki length of the key.

Description

Returns the Mki length of the key.

10.1.1.3.5 - GetMkiValue

10.1.1.3.5.1 - CCryptoKeyParam::GetMkiValue Method Deprecated since 1.7.6

Gets the mki value.

C++

```
int GetMkiValue() const;
```

Returns

The Mki value of the key.

Description

Returns the Mki value of the key. Note that this function is limited. If the Mki length is different than four bytes (sizeof(int)), the value can be truncated as it can't fit more than 4 bytes in an int. It is for this reason the function has been deprecated. The function GetMkiValue(INOUT uint8_t* puMkiValue, IN int nLength) should be used instead.

10.1.1.3.5.2 - CCryptoKeyParam::GetMkiValue Method

Gets the mki value of the key as a byte array.

puMkiValue: Array of bytes that contains the Mki value encoded as a positive decimal integer.

uLength: Length of the puMkiValue array. It should be in the range 0..128. It should be the one returned by GetMkiLength (see page 67).

C++

```
void GetMkiValue(INOUT uint8_t* puMkiValue, IN unsigned int uLenght) const;
```

Description

Returns the Mki value of the key as a byte array. If the uLength byte array is not large enough, only the first nLength bytes will be copied.

10.1.1.3.6 - CCryptoKeyParam::Parse Method

Parses the parameters list beginning at rpszStartPosition.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.

`OUT mxt_result& rres`	result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for the field KeyParam. An error is signaled in 'rres' if the data couldn't be parsed.

draft-ietf-mmusic-sdescriptions-09 ABNF: key-param = key-method ":" key-info key-method = srtp-key-method srtp-key-method = "inline" key-info = srtp-key-info srtp-key-info = key-salt ["|" lifetime] ["|" mki]

key-salt = 1*(base64) ; binary key and salt values ; concatenated together, and then ; base64 encoded [section 6.8 of ; RFC2046]

lifetime = ["2^"] 1*(DIGIT) ; see section 5.1 for "2^" mki = mki-value ":" mki-length mki-value = 1*DIGIT mki-length = 1*3DIGIT ; range 1..128.

base64 = ALPHA / DIGIT / "+" / "/" / "="

10.1.1.3.7 - CCryptoKeyParam::Reset Method

Resets this object.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 67).

10.1.1.3.8 - CCryptoKeyParam::Serialize Method

Inserts cSeparator after each parameter, except for last parameter.

C++

```
void Serialize(INOUT CBlob& rBlob);
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.1.3.9 - CCryptoKeyParam::SetKey Method

Sets the crypto key.

C++

```
void SetKey(IN const char* pszKey);
```

Parameters

Parameters	Description
`IN const char* pszKey`	Key to set. The key and salt key should be concatenated and base64 encoded

Description

This method sets the key(s) to use.

10.1.1.3.10 - CCryptoKeyParam::SetKeyMethod Method

Sets the key method.

C++

```
void SetKeyMethod(IN const char* pszKeyMethod);
```

Parameters

Parameters	Description
`IN const char* pszKeyMethod`	Key method to set

Description

This method sets the key method to use

10.1.1.3.11 - CCryptoKeyParam::SetLifeTime Method

Sets the lifetime.

C++

```
void SetLifeTime(IN uint64_t uLifeTime);
```

Parameters

Parameters	Description
`IN uint64_t uLifeTime`	Life time of the key.

Description

This method sets the life time of the key.

10.1.1.3.12 - CCryptoKeyParam::SetMki Method

Sets the mki value of the key and its length.

C++

```
void SetMki(IN uint8_t* puMkiValue, IN unsigned int uLenght);
```

Parameters

Parameters	Description
`IN uint8_t* puMkiValue`	Array of bytes that contains the Mki value encoded as a positive decimal integer.
`uLength`	Length of the array that contains the Mki length of the key. It should be in the range 0..128.

Description

This method sets the Mki value of the key and its length. The values contained in the array MUST be in network byte order.

10.1.1.3.13 - CCryptoKeyParam::SetMkiLength Method

Sets the mki length.

C++

```
void SetMkiLength(IN int nMkiLength);
```

Parameters

Parameters	Description
`IN int nMkiLength`	The Mki length of the key.

Description

This method sets the Mki length of the key. It should be in the range 0..128.

10.1.1.3.14 - CCryptoKeyParam::SetMkiValue Method Deprecated since 1.7.6

Sets the mki value.

C++

```
void SetMkiValue(IN int nMkiValue);
```

Parameters

Parameters	Description
`uMkiValue`	Mki value of the key.

Description

This method sets the Mki value of the key. Note that this function limits the Mki max length to 4 bytes, sizeof(int). For this reason the function has been deprecated and SetMki (see page 69) should be used.

10.1.1.3.15 - CCryptoKeyParam::Validate Method

Return true if data members are valid

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.1.4 - Operators

10.1.1.4.1 - CCryptoKeyParam::= Operator

Assignment operator.

C++

```
CCryptoKeyParam& operator =(IN const CCryptoKeyParam& rFrom);
```

Parameters

Parameters	Description
`IN const CCryptoKeyParam& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.1.4.2 - CCryptoKeyParam::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CCryptoKeyParam& rFrom) const;
```

Returns

true if attributes are identical.

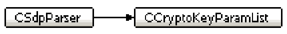
Description

Comparison operator

10.1.2 - CCryptoKeyParamList Class

This class implements a list of key parameters.

Class Hierarchy



C++

```
class CCryptoKeyParamList : public CSdpParser;
```

Description

This class implements a list of key parameters, separated by a specified character (;).

```
     draft-ietf-mmusic-sdescriptions-09 ABNF
        key-params       = key-param *(";" key-param)
```

Location

SdpParser/CCryptoKeyParamList.h

See Also

CCryptoKeyParam (see page 64)

Constructors

Constructor	Description
◆ CCryptoKeyParamList (see page 72)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CCryptoKeyParamList (see page 72)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ [] (see page 74)	Returns the parameter at uIndex.
◆ = (see page 75)	Assignment operator.
◆ == (see page 75)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ Append (see page 72)	Adds a parameter at the end of the list. Does not check if the parameter name is already in the list.
◆ IsEmpty (see page 73)	Returns true if the list contains no parameters.
◆ Length (see page 73)	Returns the number of parameters into the list.
◆ Parse (see page 73)	Parses the parameters list beginning at rpszStartPosition.
◆ Reset (see page 74)	Resets this object.
◆ Serialize (see page 74)	Inserts cSeparator after each parameter, except for last parameter.
◆ Validate (see page 74)	Returns true if data members are valid

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
Reset (see page 353)	Resets the data in the parser.
Validate (see page 353)	Validates the parsed data.

Legend

	Method
	abstract
	virtual

10.1.2.1 - Constructors

10.1.2.1.1 - CCryptoKeyParamList

10.1.2.1.1.1 - CCryptoKeyParamList::CCryptoKeyParamList Constructor

Default constructor.

C++

```
CCryptoKeyParamList();
```

Description

Default constructor.

10.1.2.1.1.2 - CCryptoKeyParamList::CCryptoKeyParamList Constructor

Copy constructor.

C++

```
CCryptoKeyParamList(IN const CCryptoKeyParamList& rSrc);
```

Parameters

Parameters	Description
`IN const CCryptoKeyParamList& rSrc`	List to copy.

Description

Copy constructor.

10.1.2.2 - Destructors

10.1.2.2.1 - CCryptoKeyParamList::~CCryptoKeyParamList Destructor

Destructor.

C++

```
virtual ~CCryptoKeyParamList();
```

Description

Destructor.

10.1.2.3 - Methods

10.1.2.3.1 - CCryptoKeyParamList::Append Method

Adds a parameter at the end of the list. Does not check if the parameter name is already in the list.

C++

```
uint32_t Append(IN TO CCryptoKeyParam* pParam);
```

Parameters

Parameters	Description
`IN TO CCryptoKeyParam* pParam`	Parameter to add to the list. Ownership is taken.

Returns

The number of parameters in the list including the new addition.

Description

This method adds a parameter at the end of the list. It does not verify if the parameter name is unique in the list.

10.1.2.3.2 - CCryptoKeyParamList::IsEmpty Method

Returns true if the list contains no parameters.

C++

```
bool IsEmpty() const;
```

Returns

True if the param list contains no parameters.

Description

Returns true if the param list contains no parameters.

10.1.2.3.3 - CCryptoKeyParamList::Length Method

Returns the number of parameters into the list.

C++

```
uint32_t Length() const;
```

Returns

The number of parameters in the list.

Description

Returns the number of parameters in the list.

10.1.2.3.4 - CCryptoKeyParamList::Parse Method

Parses the parameters list beginning at rpszStartPosition.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

This method iteratively parses a list of Key parameters separated by ';' character. This method allocates CCryptoKeyParam (see page 64) on the heap, and uses them to parse each parameter. The parsed parameters then populate the m_vpParam member. An error is signaled in 'rres' if the data couldn't be parsed.

10.1.2.3.5 - CCryptoKeyParamList::Reset Method

Resets this object.

C++

```
void Reset();
```

Description

Clears the parameter list.

10.1.2.3.6 - CCryptoKeyParamList::Serialize Method

Inserts cSeparator after each parameter, except for last parameter.

C++

```
void Serialize(INOUT CBlob& rBlob, IN char cSeparator = ';') const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	Where to output the parameter list.
`IN char cSeparator = ';'`	Separator to output between parameters.

Description

Outputs all parameters in the list. The output starts with the first parameter (no separator is output before the first param). No separator is added after the last parameter.

10.1.2.3.7 - CCryptoKeyParamList::Validate Method

Returns true if data members are valid

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.2.4 - Operators

10.1.2.4.1 - []

10.1.2.4.1.1 - CCryptoKeyParamList::[] Operator

Returns the parameter at uIndex.

C++

```
CCryptoKeyParam* operator [](IN uint32_t uIndex);
const CCryptoKeyParam* operator [](IN uint32_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint32_t uIndex`	Zero-based index of the parameter to fetch.

Returns

Pointer to the parameter at the given index. NULL if the index is out-of-bounds.

Description

This method provides array-style access to individual parameters in the list.

10.1.2.4.2 - CCryptoKeyParamList::= Operator

Assignment operator.

C++

```
CCryptoKeyParamList& operator =(IN const CCryptoKeyParamList& rFrom);
```

Parameters

Parameters	Description
`IN const CCryptoKeyParamList& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.2.4.3 - CCryptoKeyParamList::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CCryptoKeyParamList& rFrom) const;
```

Returns

true if attributes are identical.

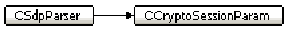
Description

Comparison operator

10.1.3 - CCryptoSessionParam Class

This class implements session parameters.

Class Hierarchy



C++

```
class CCryptoSessionParam : public CSdpParser;
```

Description

This class implements session parameters, separated by space. The ABNF of the draft-ietf-mmusic-sdescriptions-09 have been simplified. No validation is made on the multiple possibilities of parameters as specified in the draft. This is the responsibility of the users to validate the parameters.

```
     draft-ietf-mmusic-sdescriptions-09 ABNF:

         token [ EQUAL gen-value ]
         token = 1*(VCHAR)  ;visible chars [RFC2234]
         gen-value = 1*(VCHAR)  ;visible chars [RFC2234]
```

Location

SdpParser/CCryptoSessionParam.h

Constructors

Constructor	Description
CCryptoSessionParam (see page 77)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ ~CCryptoSessionParam (see page 77)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 79)	Assignment operator.
◆ == (see page 79)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetName (see page 77)	Gets the name of the key.
◆ GetValue (see page 77)	Gets the value of the key.
◆ Parse (see page 78)	Parses the parameters list beginning at rpszStartPosition. cSeparator is the character that separates the parameters.
◆ Reset (see page 78)	Resets this object.
◆ Serialize (see page 78)	Inserts cSeparator after each parameter, except for last parameter.
◆ SetName (see page 78)	Sets the name of the key.
◆ SetValue (see page 79)	Sets the value of the key.
◆ Validate (see page 79)	Returns true if data members are valid

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.3.1 - Constructors

10.1.3.1.1 - CCryptoSessionParam

10.1.3.1.1.1 - CCryptoSessionParam::CCryptoSessionParam Constructor

Default constructor.

C++

```
CCryptoSessionParam();
```

Description

Default constructor.

10.1.3.1.1.2 - CCryptoSessionParam::CCryptoSessionParam Constructor

Copy constructor.

C++

```
CCryptoSessionParam(IN const CCryptoSessionParam& rCryptoSessionParam);
```

Parameters

Parameters	Description
`rSrc`	List to copy.

Description

Copy constructor.

10.1.3.2 - Destructors

10.1.3.2.1 - CCryptoSessionParam::~CCryptoSessionParam Destructor

Destructor.

C++

```
~CCryptoSessionParam();
```

Description

Destructor.

10.1.3.3 - Methods

10.1.3.3.1 - CCryptoSessionParam::GetName Method

Gets the name of the key.

C++

```
const char* GetName() const;
```

Returns

The Name of the key.

Description

Returns the Name of the key.

10.1.3.3.2 - CCryptoSessionParam::GetValue Method

Gets the value of the key.

C++

```
const char* GetValue() const;
```

Returns

The Value of the key.

Description

Returns the Value of the key.

10.1.3.3.3 - CCryptoSessionParam::Parse Method

Parses the parameters list beginning at rpszStartPosition. cSeparator is the character that separates the parameters.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for the field SessionParam. An error is signaled in 'rres' if the data couldn't be parsed.

10.1.3.3.4 - CCryptoSessionParam::Reset Method

Resets this object.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 78).

10.1.3.3.5 - CCryptoSessionParam::Serialize Method

Inserts cSeparator after each parameter, except for last parameter.

C++

```
const void Serialize(CBlob& rBlob);
```

Parameters

Parameters	Description
`CBlob& rBlob`	The CBlob object where the data is stored.
`rData`	The object where the data is read.

Returns

None

Description

Generates the data blob from the data members.

10.1.3.3.6 - CCryptoSessionParam::SetName Method

Sets the name of the key.

C++

```
void SetName(IN const char* pszKey);
```

Parameters

Parameters	Description
`IN const char* pszKey`	Name of the key.

Returns

None.

Description

This method sets the Name of the key.

10.1.3.3.7 - CCryptoSessionParam::SetValue Method

Sets the value of the key.

C++

```
void SetValue(IN const char* pszValue);
```

Parameters

Parameters	Description
`IN const char* pszValue`	Value of the key.

Returns

None.

Description

This method sets the Value of the key.

10.1.3.3.8 - CCryptoSessionParam::Validate Method

Returns true if data members are valid

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.3.4 - Operators

10.1.3.4.1 - CCryptoSessionParam::= Operator

Assignment operator.

C++

```
CCryptoSessionParam& operator =(IN const CCryptoSessionParam& rFrom);
```

Parameters

Parameters	Description
`IN const CCryptoSessionParam& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.3.4.2 - CCryptoSessionParam::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CCryptoSessionParam& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.4 - CCryptoSessionParamList Class

This class implements a list of Session parameters.

Class Hierarchy

CSdpParser
CCryptoSessionParamList

C++

```
class CCryptoSessionParamList : public CSdpParser;
```

Description

This class implements a list of Session parameters, separated by space.

```
     draft-ietf-mmusic-sdescriptions-09 ABNF:
         *(1*WSP session-param)
```

Location

SdpParser/CCryptoSessionParamList.h

See Also

CSdpFieldAttributeCryptoKeyParam.h

Constructors

Constructor	Description
◆ CCryptoSessionParamList (see page 81)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CCryptoSessionParamList (see page 82)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ [] (see page 84)	Returns parameter at uIndex.
◆ = (see page 84)	Assignment operator.
◆ == (see page 84)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ Append (see page 82)	Adds a parameter at the end of the list. Does not check if the parameter name is already in the list.
◆ IsEmpty (see page 82)	Returns true if the list contains no parameters.
◆ Length (see page 82)	Returns the number of parameters in the list.
◆ Parse (see page 82)	Parses the parameters list beginning at rpszPos. cSeparator is the character that separates the parameters.
◆ Reset (see page 83)	Resets this object.
◆ Serialize (see page 83)	Inserts cSeparator after each parameter, except for last parameter.
◆ Validate (see page 83)	Returns true if data members are valid

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.4.1 - Constructors

10.1.4.1.1 - CCryptoSessionParamList

10.1.4.1.1.1 - CCryptoSessionParamList::CCryptoSessionParamList Constructor

Default constructor.

C++

```
CCryptoSessionParamList();
```

Description

Default constructor.

10.1.4.1.1.2 - CCryptoSessionParamList::CCryptoSessionParamList Constructor

Copy constructor.

C++

```
CCryptoSessionParamList(IN const CCryptoSessionParamList& rSrc);
```

Parameters

Parameters	Description
`IN const CCryptoSessionParamList& rSrc`	List to copy.

Description

Copy constructor.

10.1.4.2 - Destructors

10.1.4.2.1 - CCryptoSessionParamList::~CCryptoSessionParamList Destructor

Destructor.

C++

```
virtual ~CCryptoSessionParamList();
```

Description

Destructor.

10.1.4.3 - Methods

10.1.4.3.1 - CCryptoSessionParamList::Append Method

Adds a parameter at the end of the list. Does not check if the parameter name is already in the list.

C++

```
uint32_t Append(IN TO CCryptoSessionParam* pParam);
```

Parameters

Parameters	Description
`IN TO CCryptoSessionParam* pParam`	Parameter to add to the list. Ownership is taken.

Returns

The number of parameters in the list including the new addition.

Description

This method adds a parameter at the end of the list. It does not verify if the parameter name is unique in the list.

10.1.4.3.2 - CCryptoSessionParamList::IsEmpty Method

Returns true if the list contains no parameters.

C++

```
bool IsEmpty() const;
```

Returns

True if the param list contains no parameters.

Description

Returns true if the param list contains no parameters.

10.1.4.3.3 - CCryptoSessionParamList::Length Method

Returns the number of parameters in the list.

C++

```
uint32_t Length() const;
```

Returns

The number of parameters in the list.

Description

Returns the number of parameters in the list.

10.1.4.3.4 - CCryptoSessionParamList::Parse Method

Parses the parameters list beginning at rpszPos. cSeparator is the character that separates the parameters.

C++

```
CSdpParser::EParserResult Parse(INOUT const char*& rpszPos, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszPos`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

This method iteratively parses a list of sesion parameters separated by space character. This method allocates CCryptoSessionParam (see page 75) on the heap, and uses them to parse each parameter. The parsed parameters then populate the m_vpParam member. An error is signaled in 'rres' if the data couldn't be parsed.

10.1.4.3.5 - CCryptoSessionParamList::Reset Method

Resets this object.

C++

```
void Reset();
```

Description

Clears the parameter list.

10.1.4.3.6 - CCryptoSessionParamList::Serialize Method

Inserts cSeparator after each parameter, except for last parameter.

C++

```
void Serialize(INOUT CBlob& rBlob, IN char cSeparator = ';') const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	Where to output the parameter list.
`IN char cSeparator = ';'`	Separator to output between parameters.

Description

Outputs all parameters in the list. The output starts with the first parameter (no separator is output before the first param). No separator is added after the last parameter.

10.1.4.3.7 - CCryptoSessionParamList::Validate Method

Returns true if data members are valid

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.4.4 - Operators

10.1.4.4.1 - []

10.1.4.4.1.1 - CCryptoSessionParamList::[] Operator

Returns parameter at uIndex.

C++

```
CCryptoSessionParam* operator [](IN uint32_t uIndex);
const CCryptoSessionParam* operator [](IN uint32_t uIndex) const;
```

Parameters

Parameters	Description
IN uint32_t uIndex	Zero-based index of the parameter to fetch.

Returns

Pointer to the parameter at the given index. NULL if the index is out-of-bounds.

Description

This method provides array-style access to individual parameters in the list.

10.1.4.4.2 - CCryptoSessionParamList::= Operator

Assignment operator.

C++

```
CCryptoSessionParamList& operator =(IN const CCryptoSessionParamList& rFrom);
```

Parameters

Parameters	Description
IN const CCryptoSessionParamList& rFrom	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.4.4.3 - CCryptoSessionParamList::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CCryptoSessionParamList& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.5 - CSdpFieldAttributeCrypto Class

This class implements an abstraction of an attribute-crypto.

Class Hierarchy

CSdpParser
CSdpFieldAttributeCrypto

C++

```
class CSdpFieldAttributeCrypto : public CSdpParser;
```

Description

This class is an abstraction of an attribute-crypto in a SDP packet. It follows the BNF notation described in the draft-ietf-mmusic-sdescriptions-09.

```
draft-ietf-mmusic-sdescriptions-09 ABNF:

"crypto:" tag 1*WSP crypto-suite 1*WSP key-params *(1*WSP session-param)

tag            = 1*ALPHANUM
crypto-suite   = 1*(ALPHA / DIGIT / "_")

key-params     = key-param *(";" key-param)
key-param      = key-method ":" key-info
key-method     = "inline" / key-method-ext
key-method-ext = 1*(ALPHA / DIGIT / "_")
key-info       = %x21-3A / %x3C-7E ; visible (printing) characters
                                   ; except semi-colon
session-param  = 1*(VCHAR)         ; visible (printing) characters
```

Location

SdpParser/CSdpFieldAttributeCrypto.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeCrypto (↗see page 86)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (↗see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldAttributeCrypto (↗see page 86)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (↗see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (↗see page 89)	Assignment operator.
◆ == (↗see page 89)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (↗see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetCryptoSuite (↗see page 87)	Gets the CryptoSuite name.
◆ GetKeyParams (↗see page 87)	Gets the KeyParamList.
◆ GetSessionParams (↗see page 87)	Gets the SessionParamList.
◆ GetTag (↗see page 87)	Gets the Tag name of the key.
◆ Parse (↗see page 87)	Parses all the needed information for this field.
◆ Reset (↗see page 88)	Resets all the data member.

Serialize (see page 88)	Generates the data blob from the data members.
SetCryptoSuite (see page 88)	Sets the CryptoSuite name.
SetTag (see page 88)	Sets the Tag name.
Validate (see page 89)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
Reset (see page 353)	Resets the data in the parser.
Validate (see page 353)	Validates the parsed data.

Legend

	Method
A	abstract
V	virtual

10.1.5.1 - Constructors

10.1.5.1.1 - CSdpFieldAttributeCrypto

10.1.5.1.1.1 - CSdpFieldAttributeCrypto::CSdpFieldAttributeCrypto Constructor

Default constructor.

C++

```
CSdpFieldAttributeCrypto();
```

Description

Default Constructor

10.1.5.1.1.2 - CSdpFieldAttributeCrypto::CSdpFieldAttributeCrypto Constructor

Copy constructor.

C++

```
CSdpFieldAttributeCrypto(IN const CSdpFieldAttributeCrypto& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeCrypto& rFrom`	The object to be copied.

Description

Copy constructor

10.1.5.2 - Destructors

10.1.5.2.1 - CSdpFieldAttributeCrypto::~CSdpFieldAttributeCrypto Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeCrypto();
```

Description

Destructor

10.1.5.3 - Methods

10.1.5.3.1 - CSdpFieldAttributeCrypto::GetCryptoSuite Method

Gets the CryptoSuite name.

C++

```
const char* GetCryptoSuite() const;
```

Returns

The CryptoSuite name.

Description

Returns the CryptoSuite name.

10.1.5.3.2 - GetKeyParams

10.1.5.3.2.1 - CSdpFieldAttributeCrypto::GetKeyParams Method

Gets the KeyParamList.

C++

```
CCryptoKeyParamList* GetKeyParams();
const CCryptoKeyParamList* GetKeyParams() const;
```

Returns

KeyParamsList*.

Description

Returns a pointer to the KeyParamList.

10.1.5.3.3 - GetSessionParams

10.1.5.3.3.1 - CSdpFieldAttributeCrypto::GetSessionParams Method

Gets the SessionParamList.

C++

```
CCryptoSessionParamList* GetSessionParams();
const CCryptoSessionParamList* GetSessionParams() const;
```

Returns

SessionParamList*.

Description

Returns a pointer to the SessionParamList.

10.1.5.3.4 - CSdpFieldAttributeCrypto::GetTag Method

Gets the Tag name of the key.

C++

```
const char* GetTag() const;
```

Returns

The Tag name of the key.

Description

Returns the Tag name of the key.

10.1.5.3.5 - CSdpFieldAttributeCrypto::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.5.3.6 - CSdpFieldAttributeCrypto::Reset Method

Resets all the data member.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 87).

10.1.5.3.7 - CSdpFieldAttributeCrypto::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.5.3.8 - CSdpFieldAttributeCrypto::SetCryptoSuite Method

Sets the CryptoSuite name.

C++

```
void SetCryptoSuite(IN const char* pszCyptoSuite);
```

Parameters

Parameters	Description
`IN const char* pszCyptoSuite`	CryptoSuite.

Returns

None.

Description

This method sets the CryptoSuite name.

10.1.5.3.9 - CSdpFieldAttributeCrypto::SetTag Method

Sets the Tag name.

C++

```
void SetTag(IN const char* pszTag);
```

Parameters

Parameters	Description
`IN const char* pszTag`	Name of the key.

Returns

None.

Description

This method sets the Tag name.

10.1.5.3.10 - CSdpFieldAttributeCrypto::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.5.4 - Operators

10.1.5.4.1 - CSdpFieldAttributeCrypto::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeCrypto& operator =(IN const CSdpFieldAttributeCrypto& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeCrypto& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.5.4.2 - CSdpFieldAttributeCrypto::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeCrypto& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.6 - CSdpFieldAttributeFillBitRemoval Class

This class implements an abstraction of an attribute-fill-bit-removal.

Class Hierarchy

CSdpParser
CSdpFieldAttributeFillBitRemoval

C++

```
class CSdpFieldAttributeFillBitRemoval : public CSdpParser;
```

Description

This class is an abstraction of an attribute-fill-bit-removal in a SDP packet.

The parsing of this attribute-fill-bit-removal is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-fill-bit-removal  =   "T38FaxFillBitRemoval:" [bit]
bit                         =   space "0" / "1"
```

Location

SdpParser/CSdpFieldAttributeFillBitRemoval.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeFillBitRemoval (see page 91)	Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeFillBitRemoval (see page 92)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 94)	Assignment operator.
◆ == (see page 95)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ IsFillBitRemoval (see page 92)	Indicates if the T38FillBitRemoval attribute is enabled or disabled. CSdpFieldAttributeFillBitRemoval::IsFillBitRemoval

◆ IsImplicitFillBitRemoval (see page 92)	Indicates if the T38FillBitRemoval attribute is encoded with the implicit method or the explicit method. CSdpFieldAttributeFillBitRemoval::IsImplicitFillBitRemoval
◆ Parse (see page 92)	Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.
◆ Reset (see page 93)	Resets all the data members, to be ready for another call to Parse (see page 92). Disables the T38FillBitRemoval attribute. Sets the encoding method to implicit.
◆ Serialize (see page 93)	Generates the data blob from the data members.
◆ SetExplicitFillBitRemoval (see page 93)	Enables or disables the T38FillBitRemoval attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 94)(bool) and SetFillBitRemoval (see page 93)(bool) methods. CSdpFieldAttributeFillBitRemoval::SetExplicitFillBitRemoval
◆ SetFillBitRemoval (see page 93)	Enables the T38FillBitRemoval attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetFillBitRemoval(bool) method. CSdpFieldAttributeFillBitRemoval::SetFillBitRemoval
◆ SetImplicitEncoding (see page 94)	Sets the encoding method for the T38FillBitRemoval attribute. CSdpFieldAttributeFillBitRemoval::SetImplicitEncoding
◆ Validate (see page 94)	Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.6.1 - Constructors

10.1.6.1.1 - CSdpFieldAttributeFillBitRemoval

10.1.6.1.1.1 - CSdpFieldAttributeFillBitRemoval::CSdpFieldAttributeFillBitRemoval Constructor

Constructor.

C++

```
CSdpFieldAttributeFillBitRemoval();
```

Description

Default constructor.

10.1.6.1.1.2 - CSdpFieldAttributeFillBitRemoval::CSdpFieldAttributeFillBitRemoval Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeFillBitRemoval(IN const CSdpFieldAttributeFillBitRemoval& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeFillBitRemoval& rFrom`	The object to be copied.

Description

Copy constructor.

10.1.6.2 - Destructors

10.1.6.2.1 - CSdpFieldAttributeFillBitRemoval::~CSdpFieldAttributeFillBitRemoval Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeFillBitRemoval();
```

Description

Destructor

10.1.6.3 - Methods

10.1.6.3.1 - CSdpFieldAttributeFillBitRemoval::IsFillBitRemoval Method

Indicates if the T38FillBitRemoval attribute is enabled or disabled.

CSdpFieldAttributeFillBitRemoval::IsFillBitRemoval

C++

```
bool IsFillBitRemoval() const;
```

Returns

True if the T38FillBitRemoval attribute is enabled. False otherwise.

Description

Indicates if the T38FillBitRemoval attribute is enabled or disabled.

10.1.6.3.2 - CSdpFieldAttributeFillBitRemoval::IsImplicitFillBitRemoval Method

Indicates if the T38FillBitRemoval attribute is encoded with the implicit method or the explicit method.

CSdpFieldAttributeFillBitRemoval::IsImplicitFillBitRemoval

C++

```
bool IsImplicitFillBitRemoval() const;
```

Returns

True if the T38FillBitRemoval attribute is encoded with the implicit method. False if the T38FillBitRemoval attribute is encoded with the explicit method.

Description

Indicates if the T38FillBitRemoval attribute is encoded with the implicit method or the explicit method.

10.1.6.3.3 - CSdpFieldAttributeFillBitRemoval::Parse Method

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed. rres Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

Compatibility: For backward compatibility, the value string can be present or not.

Example

a=T38FaxFillBitRemoval <= Is supported and considered to be true a=T38FaxFillBitRemoval:0 <= Is supported and considered to be false a=T38FaxFillBitRemoval:1 <= Is supported and considered to be true

10.1.6.3.4 - CSdpFieldAttributeFillBitRemoval::Reset Method

Resets all the data members, to be ready for another call to Parse (see page 92). Disables the T38FillBitRemoval attribute. Sets the encoding method to implicit.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 92). Disables the T38FaxFillBitRemoval attribute. Sets the encoding method to implicit.

10.1.6.3.5 - CSdpFieldAttributeFillBitRemoval::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.6.3.6 - CSdpFieldAttributeFillBitRemoval::SetExplicitFillBitRemoval Method

Enables or disables the T38FillBitRemoval attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 94)(bool) and SetFillBitRemoval (see page 93)(bool) methods.

CSdpFieldAttributeFillBitRemoval::SetExplicitFillBitRemoval

C++

```
void SetExplicitFillBitRemoval(IN bool bSupported);
```

Parameters

Parameters	Description
`IN bool bSupported`	Indicates if the T38FillBitRemoval attribute is enabled or disabled.

Description

Enables or disables the T38FillBitRemoval attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 94)(bool) and SetFillBitRemoval (see page 93)(bool) methods.

10.1.6.3.7 - SetFillBitRemoval

10.1.6.3.7.1 - CSdpFieldAttributeFillBitRemoval::SetFillBitRemoval Method

Enables the T38FillBitRemoval attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetFillBitRemoval(bool) method.

CSdpFieldAttributeFillBitRemoval::SetFillBitRemoval

C++

```
void SetFillBitRemoval();
```

Description

Enables the T38FillBitRemoval attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetFillBitRemoval(bool) method.

10.1.6.3.7.2 - CSdpFieldAttributeFillBitRemoval::SetFillBitRemoval Method

Enables or disables the T38FillBitRemoval attribute.

CSdpFieldAttributeFillBitRemoval::SetFillBitRemoval

C++

```
void SetFillBitRemoval(bool bEnable);
```

Parameters

Parameters	Description
`bool bEnable`	Indicates if the T38FillBitRemoval attribute is enabled or disabled.

Description

Enables or disables the T38FillBitRemoval attribute.

10.1.6.3.8 - CSdpFieldAttributeFillBitRemoval::SetImplicitEncoding Method

Sets the encoding method for the T38FillBitRemoval attribute.

CSdpFieldAttributeFillBitRemoval::SetImplicitEncoding

C++

```
void SetImplicitEncoding(bool bImplicitEncoding);
```

Parameters

Parameters	Description
`bool bImplicitEncoding`	Indicates if the T38FillBitRemoval attribute is encoded with the implicit method (true) or the explicit method (false).

Description

Sets the encoding method for the T38FillBitRemoval attribute.

10.1.6.3.9 - CSdpFieldAttributeFillBitRemoval::Validate Method

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.6.4 - Operators

10.1.6.4.1 - CSdpFieldAttributeFillBitRemoval::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeFillBitRemoval& operator =(IN const CSdpFieldAttributeFillBitRemoval& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeFillBitRemoval& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.6.4.2 - CSdpFieldAttributeFillBitRemoval::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeFillBitRemoval& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.7 - CSdpFieldAttributeFmtp Class

This class implements an abstraction of an attribute-fmtp.

Class Hierarchy

CSdpParser
CSdpFieldAttributeFmtp

C++

```
class CSdpFieldAttributeFmtp : public CSdpParser;
```

Description

This class is an abstraction of an attribute-fmtp in a SDP packet.

The parsing of this attribute-fmtp does not follow the general BMF described for an attribute in RFC 2327. A restriction is put in the RFC on the format. It must be a format found in the media description. Hence, the format must be a token since the format in the media description are tokens.

```
attribute-fmtp            = "fmtp:" format format-specific-parameters
format                    = token
format-specific-parameters = *(byte-string)
byte-string               = 1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeFmtp.h

Constructors

Constructor	Description
● CSdpFieldAttributeFmtp (see page 97)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
● CSdpParser (see page 352)	Default constructor.

Legend

●	Method

Destructors

Destructor	Description
● ~CSdpFieldAttributeFmtp (see page 97)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 100)	Assignment Operator.
◆ == (see page 101)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆V GenerateCopy (see page 97)	Generates a copy of the current object.
◆ GetFmtpType (see page 98)	Gives the Fmtp type for which the format-specific-parameters pattern was implemented by the object. For the CSdpFieldAttributeFmtp, eFMTP_TYPE_UNKNOWN is returned.
◆ GetFormat (see page 98)	Returns the media format of the Fmtp field attribute as a string.
◆ GetMediaFormat (see page 98)	Returns the media format of the Fmtp field attribute.
◆V GetValue (see page 98)	Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.
◆V Parse (see page 98)	Parses the data. Can return any type of EParserResult.
◆V Reset (see page 99)	Resets the data in the parser.
◆ Serialize (see page 99)	Appends this object into the blob. Also adds a CRLF.
◆ SetFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆ SetMediaFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆V SetValue (see page 100)	Sets the value of the Fmtp field attribute to the string.
◆V Validate (see page 100)	Validates and checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

Enumerations

Enumeration	Description
EFmtpType (see page 101)	

10.1.7.1 - Data Members

10.1.7.1.1 - CSdpFieldAttributeFmtp::uINVALID_MEDIA_FORMAT Data Member

Invalid media format returned value in case of error in GetMediaFormat (see page 98).

C++

```
const uint32_t uINVALID_MEDIA_FORMAT;
```

10.1.7.2 - Constructors

10.1.7.2.1 - CSdpFieldAttributeFmtp

10.1.7.2.1.1 - CSdpFieldAttributeFmtp::CSdpFieldAttributeFmtp Constructor

Default constructor.

C++

```
CSdpFieldAttributeFmtp();
```

Description

Constructor

10.1.7.2.1.2 - CSdpFieldAttributeFmtp::CSdpFieldAttributeFmtp Constructor

Copy constructor.

C++

```
CSdpFieldAttributeFmtp(IN const CSdpFieldAttributeFmtp& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeFmtp& rFrom`	The object to be copied.

Description

Copy constructor

10.1.7.3 - Destructors

10.1.7.3.1 - CSdpFieldAttributeFmtp::~CSdpFieldAttributeFmtp Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeFmtp();
```

Description

Destructor

10.1.7.4 - Methods

10.1.7.4.1 - CSdpFieldAttributeFmtp::GenerateCopy Method

Generates a copy of the current object.

C++

```
virtual GO CSdpFieldAttributeFmtp* GenerateCopy() const;
```

Returns

A copy of the current CSdpFieldAttributeFmtp (see page 95).

Description

Generates a copy of the current object.

Warning

This methods gives ownership of the new object.

10.1.7.4.2 - CSdpFieldAttributeFmtp::GetFmtpType Method

Gives the Fmtp type for which the format-specific-parameters pattern was implemented by the object. For the CSdpFieldAttributeFmtp (see page 95), eFMTP_TYPE_UNKNOWN is returned.

C++

```
EFmtpType GetFmtpType() const;
```

Returns

The compression algorithm for this class.

eFMTP_TYPE_UNKNOWN if the class is the basic class.

Description

Returns the compression algorithm for which the format-specific-parameters pattern was implemented by the class.

10.1.7.4.3 - CSdpFieldAttributeFmtp::GetFormat Method

Returns the media format of the Fmtp field attribute as a string.

C++

```
const char* GetFormat() const;
```

Returns

A string containing the value of the media format or an empty string if the media format is invalid.

Description

Returns the media format into a string.

10.1.7.4.4 - CSdpFieldAttributeFmtp::GetMediaFormat Method

Returns the media format of the Fmtp field attribute.

C++

```
uint32_t GetMediaFormat() const;
```

Returns

The media format of this fmtp field attribute.

uINVALID_MEDIA_FORMAT (see page 96) if the format is invalid or non-numeric, in which case you should use the GetFormat (see page 98)() method.

Description

Returns the media format.

10.1.7.4.5 - CSdpFieldAttributeFmtp::GetValue Method

Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.

C++

```
virtual const char* GetValue() const;
```

Returns

The format-specific-parameters of the fmtp field attribute.

Description

Returns the format-specific-parameters into a string.

10.1.7.4.6 - CSdpFieldAttributeFmtp::Parse Method

Parses the data. Can return any type of EParserResult.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.7.4.7 - CSdpFieldAttributeFmtp::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 98).

10.1.7.4.8 - CSdpFieldAttributeFmtp::Serialize Method

Appends this object into the blob. Also adds a CRLF.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.7.4.9 - CSdpFieldAttributeFmtp::SetFormat Method

Sets the media format of the Fmtp field attribute.

C++

```
void SetFormat(IN const char* pszFormat);
```

Parameters

Parameters	Description
`IN const char* pszFormat`	The media format of the fmtp field attribute.

Description

Sets the media format of the fmtp field attribute from a string.

Warning

this method does NOT send an error result when the media format does not have the correct syntax.

10.1.7.4.10 - CSdpFieldAttributeFmtp::SetMediaFormat Method

Sets the media format of the Fmtp field attribute.

C++

```
void SetMediaFormat(IN uint32_t uFormat);
```

Parameters

Parameters	Description
`IN uint32_t uFormat`	The media format of the fmtp field attribute.

Description

Sets the media format of the fmtp field attribute from a number.

10.1.7.4.11 - CSdpFieldAttributeFmtp::SetValue Method

Sets the value of the Fmtp field attribute to the string.

C++

```
virtual void SetValue(IN const char* pszValue);
```

Parameters

Parameters	Description
`pszFormat`	The format-specific-parameters of the fmtp field attribute.

Description

Sets the format-specific-parameters of the fmtp field attribute from a string.

Warning

do not use this method in the child classes since they may have implemented specific set methods.

10.1.7.4.12 - CSdpFieldAttributeFmtp::Validate Method

Validates and checks the validity of the parsed data.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.7.5 - Operators

10.1.7.5.1 - CSdpFieldAttributeFmtp::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeFmtp& operator =(IN const CSdpFieldAttributeFmtp& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeFmtp& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator.

10.1.7.5.2 - CSdpFieldAttributeFmtp::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeFmtp& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.7.6 - Enumerations

10.1.7.6.1 - CSdpFieldAttributeFmtp::EFmtpType Enumeration

```
enum EFmtpType {
  eFMTP_TYPE_TEL_EVENT,
  eFMTP_TYPE_RED,
  eFMTP_TYPE_G7221,
  eFMTP_TYPE_AMR,
  eFMTP_TYPE_AMR_WB,
  eFMTP_TYPE_ISAC,
  eFMTP_TYPE_ILBC,
  eFTMP_TYPE_H264,
  eFTMP_TYPE_H263,
  eFTMP_TYPE_MP4V_ES,
  eFMTP_TYPE_UNKNOWN
};
```

Description

Fmtp attribute types (format-specific-parameters pattern implemented by the object).

Members

Members	Description
`eFMTP_TYPE_TEL_EVENT`	Telephone event pattern.
`eFMTP_TYPE_RED`	Redundancy pattern.
`eFMTP_TYPE_UNKNOWN`	Unknown pattern.

10.1.8 - CSdpFieldAttributeGroup Class

This class implements an abstraction of the group attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeGroup

C++

```
class CSdpFieldAttributeGroup : public CSdpParser;
```

Description

This class is an abstraction of the group attribute in SDP. The group field attribute is used to group together different media streams. It follows the BNF notation described in RFC 3388 and RFC 4091. As of now, only ANAT semantic is supported.

```
group-attribute    = "a=group:" semantics
                   *(space identification-tag)
semantics          = "ANAT"
```

Location

SdpParser/CSdpFieldAttributeGroup.h

Constructors

Constructor	Description
CSdpFieldAttributeGroup (see page 103)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeGroup (see page 103)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 105)	Assignment Operator.
◆ == (see page 105)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetIdentificationList (see page 103)	Gets the list of Identification in the group.
◆ GetSemantic (see page 104)	Gets the Semantic value.
◆ IsMember (see page 104)	Returns true if the Mid attribute is member of the group field.
◆ V Parse (see page 104)	Parses the data.
◆ V Reset (see page 104)	Resets the data in the parser.
◆ Serialize (see page 104)	Serializes the value into the blob.
◆ SetSemantic (see page 105)	Sets the Semantic value.
◆ V Validate (see page 105)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.8.1 - Constructors

10.1.8.1.1 - CSdpFieldAttributeGroup

10.1.8.1.1.1 - CSdpFieldAttributeGroup::CSdpFieldAttributeGroup Constructor

Default constructor.

C++

```
CSdpFieldAttributeGroup();
```

Description

Constructor

10.1.8.1.1.2 - CSdpFieldAttributeGroup::CSdpFieldAttributeGroup Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeGroup(IN const CSdpFieldAttributeGroup& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeGroup& rSrc`	The CSdpFieldAttributeGroup to be copied.

Description

Copy constructor

10.1.8.2 - Destructors

10.1.8.2.1 - CSdpFieldAttributeGroup::~CSdpFieldAttributeGroup Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeGroup();
```

Description

Destructor

10.1.8.3 - Methods

10.1.8.3.1 - GetIdentificationList

10.1.8.3.1.1 - CSdpFieldAttributeGroup::GetIdentificationList Method

Gets the list of Identification in the group.

C++

```
CList<CString>& GetIdentificationList();
```

Returns

A reference to the list of identification tags.

Description

Returns a reference to the list of identification tags.

10.1.8.3.1.2 - CSdpFieldAttributeGroup::GetIdentificationList Method

Gets the list of Identification in the group.

C++

```
const CList<CString>& GetIdentificationList() const;
```

Returns

A const reference to the list of identification tags.

Description

Returns a const reference to the list of identification tags.

10.1.8.3.2 - CSdpFieldAttributeGroup::GetSemantic Method

Gets the Semantic value.

C++

```
const char* GetSemantic() const;
```

Returns

The semantic value.

Description

Returns the semantic value.

10.1.8.3.3 - CSdpFieldAttributeGroup::IsMember Method

Returns true if the Mid attribute is member of the group field.

C++

```
bool IsMember(IN const CString& strMid) const;
```

Parameters

Parameters	Description
`IN const CString& strMid`	The Mid attribute for which to search.

Returns

True if the Mid attribute is present in the group.

Description

This method searches if the Mid attribute is present in the group.

10.1.8.3.4 - CSdpFieldAttributeGroup::Parse Method

Parses the data.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

10.1.8.3.5 - CSdpFieldAttributeGroup::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 104).

10.1.8.3.6 - CSdpFieldAttributeGroup::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.8.3.7 - CSdpFieldAttributeGroup::SetSemantic Method

Sets the Semantic value.

C++

```
void SetSemantic(IN const char* szValue);
```

Parameters

Parameters	Description
`IN const char* szValue`	The semantic value to set.

Description

Sets the semantic value.

10.1.8.3.8 - CSdpFieldAttributeGroup::Validate Method

Validates the parsed data.

C++

```
virtual bool Validate();
```

Returns

False if one of the data members is empty, true otherwise.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.8.4 - Operators

10.1.8.4.1 - CSdpFieldAttributeGroup::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeGroup& operator =(IN const CSdpFieldAttributeGroup& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeGroup& rSrc`	The CSdpFieldAttributeGroup (see page 101) to be copied.

Description

Assignment operator.

10.1.8.4.2 - CSdpFieldAttributeGroup::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeGroup& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeGroup& rFrom`	The CSdpFieldAttributeGroup (see page 101) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.9 - CSdpFieldAttributeIceCandidate Class

Implements the ice-candidate attribute.

Class Hierarchy

C++

```
class CSdpFieldAttributeIceCandidate : public CSdpParser;
```

Description

This class is an abstraction of an ice-candidate. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

From draft-ietf-mmusic-ice-19:

```
candidate-attribute   = "candidate" ":" foundation SP component-id SP
                        transport SP
                        priority SP
                        connection-address SP     ;from RFC 4566
                        port         ;port from RFC 4566
                        SP cand-type
                        [SP rel-addr]
                        [SP rel-port]
                        *(SP extension-att-name SP
                             extension-att-value)

foundation            = 1*32ice-char
component-id          = 1*5DIGIT
transport             = "UDP" / transport-extension
transport-extension   = token              ; from RFC 3261
priority              = 1*10DIGIT
cand-type             = "typ" SP candidate-types
candidate-types       = "host" / "srflx" / "prflx" / "relay" / token
rel-addr              = "raddr" SP connection-address
rel-port              = "rport" SP port
extension-att-name    = byte-string    ;from RFC 4566
extension-att-value   = byte-string
```

Location

SdpParser/CSdpFieldAttributeIceCandidate.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeIceCandidate (see page 108)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeIceCandidate (see page 108)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 113)	Assignment Operator.
◆ == (see page 114)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetCandidate (see page 108)	Gets a reference to the candidate string.
◆ GetComponentId (see page 109)	Returns the component ID.
◆ GetConnectionAddr (see page 109)	Gets the connection address.
◆ GetExtensionAttr (see page 109)	Gets a reference to the list of extension attributes.
◆ GetFoundation (see page 110)	Returns a reference to the foundation string.
◆ GetMicroLitePort (see page 110)	Gets the port from the microliteport extension.
◆ GetPriority (see page 110)	Gets the priority
◆ GetRelAddr (see page 111)	Gets the related connection address.
◆ GetTransport (see page 111)	Gets a reference to the transport string.
◆ V Parse (see page 111)	Parses all the needed information for this field.
◆ V Reset (see page 112)	Resets this object.
◆ Serialize (see page 112)	Serializes the value into the blob.
◆ SetComponentId (see page 112)	Sets the component ID.
◆ SetConnectionAddr (see page 112)	Sets the connection address.
◆ SetMicroLitePort (see page 112)	Sets the port in the microliteport extension.
◆ SetPriority (see page 113)	Sets the priority
◆ SetRelAddr (see page 113)	Sets the related connection address.
◆ V Validate (see page 113)	Returns true if data members are valid

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.9.1 - Constructors

10.1.9.1.1 - CSdpFieldAttributeIceCandidate

10.1.9.1.1.1 - CSdpFieldAttributeIceCandidate::CSdpFieldAttributeIceCandidate Constructor

Default constructor.

C++

```
CSdpFieldAttributeIceCandidate();
```

Description

Default constructor.

10.1.9.1.1.2 - CSdpFieldAttributeIceCandidate::CSdpFieldAttributeIceCandidate Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIceCandidate(IN const CSdpFieldAttributeIceCandidate& rSrc);
```

Description

Copy constructor.

10.1.9.2 - Destructors

10.1.9.2.1 - CSdpFieldAttributeIceCandidate::~CSdpFieldAttributeIceCandidate Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIceCandidate();
```

Description

Destructor.

10.1.9.3 - Methods

10.1.9.3.1 - GetCandidate

10.1.9.3.1.1 - CSdpFieldAttributeIceCandidate::GetCandidate Method

Gets a reference to the candidate string.

C++

```
CString& GetCandidate();
```

Returns

a reference to the candidate string.

Description

Returns a reference to the candidate string.

10.1.9.3.1.2 - CSdpFieldAttributeIceCandidate::GetCandidate Method

Gets a reference to the candidate string.

C++

```
const CString& GetCandidate() const;
```

Returns

a reference to the candidate string.

Description

Returns a reference to the candidate string.

10.1.9.3.2 - CSdpFieldAttributeIceCandidate::GetComponentId Method

Returns the component ID.

C++

```
uint32_t GetComponentId() const;
```

Returns

the component ID.

Description

Returns the component ID.

10.1.9.3.3 - CSdpFieldAttributeIceCandidate::GetConnectionAddr Method

Gets the connection address.

C++

```
void GetConnectionAddr(INOUT const CSocketAddr** ppAddr, INOUT const CFqdn** ppFqdnAddr) const;
```

Parameters

Parameters	Description
`INOUT const CSocketAddr** ppAddr`	Pointer of pointer to a CSocketAddr.
`INOUT const CFqdn** ppFqdnAddr`	Pointer of pointer to a CFqdn.

Description

Returns either a CSocketAddr or a CFqdn. Only one is set. If the address is an IPv4 or IPv6 address, the ppAddr is set. If the address is an FQDN, only the ppFqdnAddr is set. The user must check which one is non-NULL before using it. Also note that the returned pointer is only valid as long as the CSdpFieldAttributeIceCandidate (see page 106) exists.

10.1.9.3.4 - GetExtensionAttr

10.1.9.3.4.1 - CSdpFieldAttributeIceCandidate::GetExtensionAttr Method

Gets a reference to the list of extension attributes.

C++

```
CList<SExtensionAtt>& GetExtensionAttr();
```

Returns

a reference to the list of extension attributes.

Description

Returns a reference to the list of extension attributes.

10.1.9.3.4.2 - CSdpFieldAttributeIceCandidate::GetExtensionAttr Method

Gets a reference to the list of extension attributes.

C++

```
const CList<SExtensionAtt>& GetExtensionAttr() const;
```

Returns

a reference to the list of extension attributes.

Description

Returns a reference to the list of extension attributes.

10.1.9.3.5 - GetFoundation

10.1.9.3.5.1 - CSdpFieldAttributeIceCandidate::GetFoundation Method

Returns a reference to the foundation string.

C++

```
CString& GetFoundation();
```

Returns

a reference to the foundation.

Description

Returns a reference to the foundation string.

10.1.9.3.5.2 - CSdpFieldAttributeIceCandidate::GetFoundation Method

Returns a reference to the foundation string.

C++

```
const CString& GetFoundation() const;
```

Returns

a reference to the foundation.

Description

Returns a reference to the foundation string.

10.1.9.3.6 - CSdpFieldAttributeIceCandidate::GetMicroLitePort Method

Gets the port from the microliteport extension.

C++

```
mxt_result GetMicroLitePort(OUT uint16_t& ruPort) const;
```

Parameters

Parameters	Description
`OUT uint16_t& ruPort`	Reference to the variable where to put the port number.

Returns

- resS_OK: success.
- Other return codes: Failure.

Description

Gets the microliteport extension value. If the extension is not found, the method returns a failure.

10.1.9.3.7 - CSdpFieldAttributeIceCandidate::GetPriority Method

Gets the priority

C++

```
uint64_t GetPriority() const;
```

Returns

the priority.

Description

Returns the priority.

10.1.9.3.8 - CSdpFieldAttributeIceCandidate::GetRelAddr Method

Gets the related connection address.

C++

```
void GetRelAddr(INOUT const CSocketAddr** ppAddr, INOUT const CFqdn** ppFqdnAddr) const;
```

Parameters

Parameters	Description
`INOUT const CSocketAddr** ppAddr`	Pointer of pointer to a CSocketAddr.
`INOUT const CFqdn** ppFqdnAddr`	Pointer of pointer to a CFqdn.

Description

Returns either a CSocketAddr or a CFqdn. Only one is set. If the address is an IPv4 or IPv6 address, the ppAddr is set. If the address is an FQDN, only the ppFqdnAddr is set. The user must check which one is non-NULL before using it. Also note that the returned pointer is only valid as long as the CSdpFieldAttributeIceCandidate (see page 106) exists.

10.1.9.3.9 - GetTransport

10.1.9.3.9.1 - CSdpFieldAttributeIceCandidate::GetTransport Method

Gets a reference to the transport string.

C++

```
CString& GetTransport();
```

Returns

a reference to the transport string.

Description

Returns a reference to the transport string.

10.1.9.3.9.2 - CSdpFieldAttributeIceCandidate::GetTransport Method

Gets a reference to the transport string.

C++

```
const CString& GetTransport() const;
```

Returns

a reference to the transport string.

Description

Returns a reference to the transport string.

10.1.9.3.10 - CSdpFieldAttributeIceCandidate::Parse Method

Parses all the needed information for this field.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.9.3.11 - CSdpFieldAttributeIceCandidate::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 111).

10.1.9.3.12 - CSdpFieldAttributeIceCandidate::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string with all the set candidate parameters.

10.1.9.3.13 - CSdpFieldAttributeIceCandidate::SetComponentId Method

Sets the component ID.

C++

```
void SetComponentId(IN uint32_t uComponentId);
```

Parameters

Parameters	Description
`IN uint32_t uComponentId`	The component-ID.

Description

Sets the component-ID. It must not exeed 5 digits to be valid.

10.1.9.3.14 - CSdpFieldAttributeIceCandidate::SetConnectionAddr Method

Sets the connection address.

C++

```
void SetConnectionAddr(IN CSocketAddr addr);
```

Parameters

Parameters	Description
`IN CSocketAddr addr`	The connection address to set.

Description

Sets the connection address.

10.1.9.3.15 - CSdpFieldAttributeIceCandidate::SetMicroLitePort Method

Sets the port in the microliteport extension.

C++

```
void SetMicroLitePort(IN uint16_t uPort);
```

Parameters

Parameters	Description
`IN uint16_t uPort`	The port number to set for the candidate.

Description

Sets the microliteport extension with the specified port. Adds the microliteport extension if it is not already present in the candidate.

10.1.9.3.16 - CSdpFieldAttributeIceCandidate::SetPriority Method

Sets the priority

C++

```
void SetPriority(IN uint64_t uPriority);
```

Parameters

Parameters	Description
`IN uint64_t uPriority`	The priority to set.

Description

Sets the priority.

10.1.9.3.17 - CSdpFieldAttributeIceCandidate::SetRelAddr Method

Sets the related connection address.

C++

```
void SetRelAddr(IN CSocketAddr addr);
```

Parameters

Parameters	Description
`IN CSocketAddr addr`	The connection address to set.

Description

Sets the connection address.

10.1.9.3.18 - CSdpFieldAttributeIceCandidate::Validate Method

Returns true if data members are valid

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.9.4 - Operators

10.1.9.4.1 - CSdpFieldAttributeIceCandidate::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIceCandidate& operator =(IN const CSdpFieldAttributeIceCandidate& rSrc);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.9.4.2 - CSdpFieldAttributeIceCandidate::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIceCandidate& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.10 - CSdpFieldAttributeIceOptions Class

Implements the ice-options attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeIceOptions

C++

```
class CSdpFieldAttributeIceOptions : public CSdpParser;
```

Description

This class is an abstraction of an ice-options. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

From draft-ietf-mmusic-ice-19:

```
ice-options         = "ice-options" ":" ice-option-tag
                    0*(SP ice-option-tag)
ice-option-tag       = 1*ice-char
```

Location

SdpParser/CSdpFieldAttributeIceOptions.h

Constructors

Constructor	Description
● CSdpFieldAttributeIceOptions (see page 115)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
● CSdpParser (see page 352)	Default constructor.

Legend

●	Method

Destructors

Destructor	Description
● V ~CSdpFieldAttributeIceOptions (see page 116)	Destructor.

CSdpParser Class

CSdpParser Class	Description
● V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 117)	Assignment Operator.
◆ == (see page 117)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetOptionTagsList (see page 116)	Gets the ice-options tag list.
◆ V Parse (see page 116)	Parses all the needed information for this field.
◆ V Reset (see page 116)	Resets this object.
◆ Serialize (see page 117)	Serializes the value into the blob.
◆ V Validate (see page 117)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.10.1 - Constructors

10.1.10.1.1 - CSdpFieldAttributeIceOptions

10.1.10.1.1.1 - CSdpFieldAttributeIceOptions::CSdpFieldAttributeIceOptions Constructor

Default constructor.

C++

```
CSdpFieldAttributeIceOptions();
```

Description

Default constructor.

10.1.10.1.1.2 - CSdpFieldAttributeIceOptions::CSdpFieldAttributeIceOptions Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIceOptions(IN const CSdpFieldAttributeIceOptions& rSrc);
```

10.1.10.2 - Destructors

10.1.10.2.1 - CSdpFieldAttributeIceOptions::~CSdpFieldAttributeIceOptions Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIceOptions();
```

Description

Destructor.

10.1.10.3 - Methods

10.1.10.3.1 - GetOptionTagsList

10.1.10.3.1.1 - CSdpFieldAttributeIceOptions::GetOptionTagsList Method

Gets the ice-options tag list.

C++

```
CList<CString>& GetOptionTagsList();
```

Returns

The list of ICE options tag.

Description

Returns the list of ICE options tag.

10.1.10.3.1.2 - CSdpFieldAttributeIceOptions::GetOptionTagsList Method

```
const CList<CString>& GetOptionTagsList() const;
```

Returns

The list of ICE options tag.

Description

Returns the list of ICE options tag.

10.1.10.3.2 - CSdpFieldAttributeIceOptions::Parse Method

Parses all the needed information for this field.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.10.3.3 - CSdpFieldAttributeIceOptions::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 116).

10.1.10.3.4 - CSdpFieldAttributeIceOptions::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string from the set ICE options tags list.

10.1.10.3.5 - CSdpFieldAttributeIceOptions::Validate Method

Returns true if data members are valid.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.10.4 - Operators

10.1.10.4.1 - CSdpFieldAttributeIceOptions::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIceOptions& operator =(IN const CSdpFieldAttributeIceOptions& rSrc);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.10.4.2 - CSdpFieldAttributeIceOptions::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIceOptions& rFrom) const;
```

Returns

true if both attributes contain the same ICE option tag.

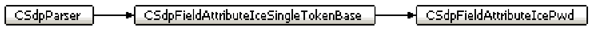
Description

Comparison operator

10.1.11 - CSdpFieldAttributeIcePwd Class

Implements the ice-pwd attribute.

Class Hierarchy



C++

```
class CSdpFieldAttributeIcePwd : public CSdpFieldAttributeIceSingleTokenBase;
```

Description

This class is an abstraction of an ice-pwd. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

From draft-ietf-mmusic-ice-19:

```
ice-pwd-att        = "ice-pwd" ":" password
password            = 22*256ice-char
```

Location

SdpParser/CSdpFieldAttributeIcePwd.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeIcePwd (see page 119)	Default Constructor.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ CSdpFieldAttributeIceSingleTokenBase (see page 129)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeIcePwd (see page 119)	Destructor.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ V ~CSdpFieldAttributeIceSingleTokenBase (see page 129)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 120)	Assignment Operator.
◆ == (see page 121)	Comparison Operator.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ = (see page 131)	Assignment Operator.
◆ == (see page 131)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetPassword (see page 120)	Gets the UserFrag.
◆ Serialize (see page 120)	Serializes the value into the blob.
◆ SetPassword (see page 120)	Sets the UserFrag.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆V Parse (see page 130)	Parses all the needed information for this field.
◆V Reset (see page 130)	Resets this object.
◆V Serialize (see page 130)	Serializes the value into the blob.
◆V Validate (see page 130)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.11.1 - Constructors

10.1.11.1.1 - CSdpFieldAttributeIcePwd

10.1.11.1.1.1 - CSdpFieldAttributeIcePwd::CSdpFieldAttributeIcePwd Constructor

Default Constructor.

C++

```
CSdpFieldAttributeIcePwd();
```

Description

Default constructor.

10.1.11.1.1.2 - CSdpFieldAttributeIcePwd::CSdpFieldAttributeIcePwd Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIcePwd(IN const CSdpFieldAttributeIcePwd& rSrc);
```

10.1.11.2 - Destructors

10.1.11.2.1 - CSdpFieldAttributeIcePwd::~CSdpFieldAttributeIcePwd Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIcePwd();
```

Description

Destructor.

10.1.11.3 - Methods

10.1.11.3.1 - CSdpFieldAttributeIcePwd::GetPassword Method

Gets the UserFrag.

C++

```
const char* GetPassword() const;
```

Returns

The password.

Description

Returns the password.

10.1.11.3.2 - CSdpFieldAttributeIcePwd::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string from the set password and appends it to the blob that is passed in reference.

10.1.11.3.3 - CSdpFieldAttributeIcePwd::SetPassword Method

Sets the UserFrag.

C++

```
void SetPassword(IN const char* pszPwd);
```

Parameters

Parameters	Description
`IN const char* pszPwd`	The password to set.

Description

Sets the password.

10.1.11.4 - Operators

10.1.11.4.1 - CSdpFieldAttributeIcePwd::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIcePwd& operator =(IN const CSdpFieldAttributeIcePwd& rSrc);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.11.4.2 - CSdpFieldAttributeIcePwd::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIcePwd& rFrom) const;
```

Returns

true if both attributes contain the same password.

Description

Comparison operator

10.1.12 - CSdpFieldAttributeIceRemoteCandidates Class

Implements the ICE remote-candidates attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeIceRemoteCandidates

C++

```
class CSdpFieldAttributeIceRemoteCandidates : public CSdpParser;
```

Description

This class is an abstraction of an ice-remote-candidates. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

From draft-ietf-mmusic-ice-19:

```
remote-candidate-att = "remote-candidates" ":" remote-candidate
                        0*(SP remote-candidate)
remote-candidate = component-ID SP connection-address SP port
```

Location

SdpParser/CSdpFieldAttributeIceRemoteCandidates.h

Classes

Class	Description
CIceRemoteCandidates (see page 122)	Container for ICE remote candidate attributes

Constructors

Constructor	Description
◆ CSdpFieldAttributeIceRemoteCandidates (see page 125)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ ▼ ~CSdpFieldAttributeIceRemoteCandidates (see page 126)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ ▼ ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (↗see page 127)	Assignment Operator.
◆ == (↗see page 128)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (↗see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetIceRemoteCandidates (↗see page 126)	Gets the ICE remote candidates vector.
◆V Parse (↗see page 126)	Parses all the needed information for this field.
◆V Reset (↗see page 127)	Resets this object.
◆ Serialize (↗see page 127)	Serializes the value into the blob.
◆V Validate (↗see page 127)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (↗see page 353)	Returns true if the data was parsed successfully.
◆A Parse (↗see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (↗see page 353)	Resets the data in the parser.
◆A Validate (↗see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.12.1 - Classes

10.1.12.1.1 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates Class

Container for ICE remote candidate attributes

Class Hierarchy

C++

```
class CIceRemoteCandidates;
```

Constructors

Constructor	Description
◆ CIceRemoteCandidates (↗see page 123)	Default Constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CIceRemoteCandidates (↗see page 123)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 125)	Assignment Operator.
◆ == (see page 125)	Comparison Operator.

Legend

◆	Method

Methods

Method	Description
◆ GetComponentId (see page 124)	Returns the component ID.
◆ GetConnectionAddr (see page 124)	Gets the connection address.
◆ SetComponentId (see page 124)	Sets the component ID.
◆ SetConnectionAddr (see page 124)	Sets the connection address.
◆ SetConnectionFqdn (see page 125)	Sets the connection fqdn.

Legend

◆	Method

10.1.12.1.1.1 - Constructors

10.1.12.1.1.1.1 - CIceRemoteCandidates

10.1.12.1.1.1.1.1 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::CIceRemoteCandidates Constructor

Default Constructor.

C++

```
CIceRemoteCandidates();
```

Description

Default constructor.

10.1.12.1.1.1.1.2 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::CIceRemoteCandidates Constructor

Copy Constructor.

C++

```
CIceRemoteCandidates(IN const CIceRemoteCandidates& rSrc);
```

10.1.12.1.1.2 - Destructors

10.1.12.1.1.2.1 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::~CIceRemoteCandidates Destructor

Destructor.

C++

```
virtual ~CIceRemoteCandidates();
```

Description

Destructor.

10.1.12.1.1.3 - Methods

10.1.12.1.1.3.1 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::GetComponentId Method

Returns the component ID.

C++

```
uint32_t GetComponentId() const;
```

Returns

the component ID.

Description

Returns the component ID.

10.1.12.1.1.3.2 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::GetConnectionAddr Method

Gets the connection address.

C++

```
void GetConnectionAddr(INOUT const CSocketAddr** ppAddr, INOUT const CFqdn** ppFqdnAddr);
```

Parameters

Parameters	Description
`INOUT const CSocketAddr** ppAddr`	Pointer of pointer to a CSocketAddr.
`INOUT const CFqdn** ppFqdnAddr`	Pointer of pointer to a CFqdn.

Description

Returns either a CSocketAddr or a CFqdn. Only one is set. If the address is an IPv4 or IPv6 address, the ppAddr is set. If the address is an FQDN, only the ppFqdnAddr is set. The user must check which one is non-NULL before using it. Also note that the returned pointer is only valid as long as the CSdpFieldAttributeIceRemoteCandidates (see page 121) exists.

10.1.12.1.1.3.3 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::SetComponentId Method

Sets the component ID.

C++

```
void SetComponentId(IN uint32_t uComponentId);
```

Parameters

Parameters	Description
`IN uint32_t uComponentId`	The component-ID.

Description

Sets the component-ID. It must not exeed 5 digits to be valid.

10.1.12.1.1.3.4 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::SetConnectionAddr Method

Sets the connection address.

C++

```
void SetConnectionAddr(IN CSocketAddr addr);
```

Parameters

Parameters	Description
`IN CSocketAddr addr`	The connection address to set.

Description

Sets the connection address.

10.1.12.1.1.3.5 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::SetConnectionFqdn Method

Sets the connection fqdn.

C++

```
void SetConnectionFqdn(IN CFqdn fqdn);
```

Parameters

Parameters	Description
`IN CFqdn fqdn`	The connection FQDN to set.

Description

Sets the connection FQDN.

10.1.12.1.1.4 - Operators

10.1.12.1.1.4.1 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::= Operator

Assignment Operator.

C++

```
CIceRemoteCandidates& operator =(IN const CIceRemoteCandidates& rFrom);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.12.1.1.4.2 - CSdpFieldAttributeIceRemoteCandidates::CIceRemoteCandidates::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CIceRemoteCandidates& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.12.1.1.5 - Friends

10.1.12.1.1.5.1 - friend class CSdpFieldAttributeIceRemoteCandidates Friend

```
friend class CSdpFieldAttributeIceRemoteCandidates;
```

10.1.12.2 - Constructors

10.1.12.2.1 - CSdpFieldAttributeIceRemoteCandidates

10.1.12.2.1.1 - CSdpFieldAttributeIceRemoteCandidates::CSdpFieldAttributeIceRemoteCandidates Constructor

Default constructor.

C++

```
CSdpFieldAttributeIceRemoteCandidates();
```

Description

Default constructor.

10.1.12.2.1.2 - CSdpFieldAttributeIceRemoteCandidates::CSdpFieldAttributeIceRemoteCandidates Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIceRemoteCandidates(IN const CSdpFieldAttributeIceRemoteCandidates& rSrc);
```

Description

Copy constructor.

10.1.12.3 - Destructors

10.1.12.3.1 - CSdpFieldAttributeIceRemoteCandidates::~CSdpFieldAttributeIceRemoteCandidates Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIceRemoteCandidates();
```

Description

Destructor.

10.1.12.4 - Methods

10.1.12.4.1 - GetIceRemoteCandidates

10.1.12.4.1.1 - CSdpFieldAttributeIceRemoteCandidates::GetIceRemoteCandidates Method

Gets the ICE remote candidates vector.

C++

```
CVector<CIceRemoteCandidates*>& GetIceRemoteCandidates();
```

Returns

The CIceRemoteCandidates (see page 122) vector.

Description

Returns the CIceRemoteCandidates (see page 122) vector.

10.1.12.4.1.2 - CSdpFieldAttributeIceRemoteCandidates::GetIceRemoteCandidates Method

```
const CVector<CIceRemoteCandidates*>& GetIceRemoteCandidates() const;
```

Returns

The CIceRemoteCandidates (see page 122) vector.

Description

Returns the CIceRemoteCandidates (see page 122) vector.

10.1.12.4.2 - CSdpFieldAttributeIceRemoteCandidates::Parse Method

Parses all the needed information for this field.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.12.4.3 - CSdpFieldAttributeIceRemoteCandidates::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 126).

10.1.12.4.4 - CSdpFieldAttributeIceRemoteCandidates::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string with all the set candidate parameters.

10.1.12.4.5 - CSdpFieldAttributeIceRemoteCandidates::Validate Method

Returns true if data members are valid.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.12.5 - Operators

10.1.12.5.1 - CSdpFieldAttributeIceRemoteCandidates::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIceRemoteCandidates& operator =(IN const CSdpFieldAttributeIceRemoteCandidates& rFrom);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.12.5.2 - CSdpFieldAttributeIceRemoteCandidates::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIceRemoteCandidates& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.13 - CSdpFieldAttributeIceSingleTokenBase Class

Base class for single token ICE attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeIceSingleTokenBase

C++

```
class CSdpFieldAttributeIceSingleTokenBase : public CSdpParser;
```

Description

This base class is an abstraction of some of single token field of ICE. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

Location

SdpParser/CSdpFieldAttributeIceSingleTokenBase.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeIceSingleTokenBase (see page 129)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeIceSingleTokenBase (see page 129)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
= (see page 131)	Assignment Operator.
== (see page 131)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
= (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
Parse (see page 130)	Parses all the needed information for this field.
Reset (see page 130)	Resets this object.
Serialize (see page 130)	Serializes the value into the blob.
Validate (see page 130)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
Reset (see page 353)	Resets the data in the parser.
Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.13.1 - Constructors

10.1.13.1.1 - CSdpFieldAttributeIceSingleTokenBase

10.1.13.1.1.1 - CSdpFieldAttributeIceSingleTokenBase::CSdpFieldAttributeIceSingleTokenBase Constructor

Default constructor.

C++

```
CSdpFieldAttributeIceSingleTokenBase();
```

Description

Default constructor.

10.1.13.1.1.2 - CSdpFieldAttributeIceSingleTokenBase::CSdpFieldAttributeIceSingleTokenBase Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIceSingleTokenBase(IN const CSdpFieldAttributeIceSingleTokenBase& rSrc);
```

10.1.13.2 - Destructors

10.1.13.2.1 - CSdpFieldAttributeIceSingleTokenBase::~CSdpFieldAttributeIceSingleTokenBase Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIceSingleTokenBase();
```

Description

Destructor.

10.1.13.3 - Methods

10.1.13.3.1 - CSdpFieldAttributeIceSingleTokenBase::Parse Method

Parses all the needed information for this field.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.13.3.2 - CSdpFieldAttributeIceSingleTokenBase::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 130).

10.1.13.3.3 - CSdpFieldAttributeIceSingleTokenBase::Serialize Method

Serializes the value into the blob.

C++

```
virtual void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string from the set user fragment and appends it to the blob that is passed in reference.

10.1.13.3.4 - CSdpFieldAttributeIceSingleTokenBase::Validate Method

Returns true if data members are valid.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.

- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.13.4 - Operators

10.1.13.4.1 - CSdpFieldAttributeIceSingleTokenBase::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIceSingleTokenBase& operator =(IN const CSdpFieldAttributeIceSingleTokenBase& rSrc);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.13.4.2 - CSdpFieldAttributeIceSingleTokenBase::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIceSingleTokenBase& rFrom) const;
```

Returns

true if both attributes contain the same user fragment.

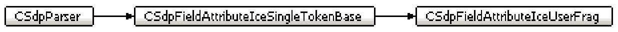
Description

Comparison operator

10.1.14 - CSdpFieldAttributeIceUserFrag Class

Implements the ice-ufrag attribute.

Class Hierarchy


CSdpParser
CSdpFieldAttributeIceSingleTokenBase
CSdpFieldAttributeIceUserFrag

C++

```
class CSdpFieldAttributeIceUserFrag : public CSdpFieldAttributeIceSingleTokenBase;
```

Description

This class is an abstraction of an ice-ufrag. It follows the BNF notation described in the draft-ietf-mmusic-ice-19.

From draft-ietf-mmusic-ice-19:

```
ice-ufrag-att        = "ice-ufrag" ":" ufrag
ufrag               = 4*256ice-char
```

Location

SdpParser/CSdpFieldAttributeIceUserFrag.h

Constructors

Constructor	Description
CSdpFieldAttributeIceUserFrag (see page 133)	Default Constructor.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
CSdpFieldAttributeIceSingleTokenBase (see page 129)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeIceUserFrag (see page 133)	Destructor.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ V ~CSdpFieldAttributeIceSingleTokenBase (see page 129)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 134)	Assignment Operator.
◆ == (see page 134)	Comparison Operator.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ = (see page 131)	Assignment Operator.
◆ == (see page 131)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetUserFrag (see page 133)	Gets the UserFrag.
◆ Serialize (see page 133)	Serializes the value into the blob.
◆ SetUserFrag (see page 134)	Sets the UserFrag.

CSdpFieldAttributeIceSingleTokenBase Class

CSdpFieldAttributeIceSingleTokenBase Class	Description
◆ V Parse (see page 130)	Parses all the needed information for this field.
◆ V Reset (see page 130)	Resets this object.
◆ V Serialize (see page 130)	Serializes the value into the blob.
◆ V Validate (see page 130)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.14.1 - Constructors

10.1.14.1.1 - CSdpFieldAttributeIceUserFrag

10.1.14.1.1.1 - CSdpFieldAttributeIceUserFrag::CSdpFieldAttributeIceUserFrag Constructor

Default Constructor.

C++

```
CSdpFieldAttributeIceUserFrag();
```

Description

Default constructor.

10.1.14.1.1.2 - CSdpFieldAttributeIceUserFrag::CSdpFieldAttributeIceUserFrag Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeIceUserFrag(IN const CSdpFieldAttributeIceUserFrag& rSrc);
```

10.1.14.2 - Destructors

10.1.14.2.1 - CSdpFieldAttributeIceUserFrag::~CSdpFieldAttributeIceUserFrag Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeIceUserFrag();
```

Description

Destructor.

10.1.14.3 - Methods

10.1.14.3.1 - CSdpFieldAttributeIceUserFrag::GetUserFrag Method

Gets the UserFrag.

C++

```
const char* GetUserFrag() const;
```

Returns

The user fragment.

Description

Returns the user fragment.

10.1.14.3.2 - CSdpFieldAttributeIceUserFrag::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob where the serialized attribute is appended.

Description

Creates a text string from the set user fragment and appends it to the blob that is passed in reference.

10.1.14.3.3 - CSdpFieldAttributeIceUserFrag::SetUserFrag Method

Sets the UserFrag.

C++

```
void SetUserFrag(IN const char* pszUserFrag);
```

Parameters

Parameters	Description
`IN const char* pszUserFrag`	The user fragment to set.

Description

Sets the user fragment.

10.1.14.4 - Operators

10.1.14.4.1 - CSdpFieldAttributeIceUserFrag::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeIceUserFrag& operator =(IN const CSdpFieldAttributeIceUserFrag& rSrc);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.14.4.2 - CSdpFieldAttributeIceUserFrag::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeIceUserFrag& rFrom) const;
```

Returns

true if both attributes contain the same user fragment.

Description

Comparison operator

10.1.15 - CSdpFieldAttributeKeyMgmt Class

This class implements an abstraction of a key-mgmt-attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeKeyMgmt

C++

```
class CSdpFieldAttributeKeyMgmt : public CSdpParser;
```

Description

This class is an abstraction of an key-mgmt-attribute in a SDP packet.

```
key-mgmt-attribute = key-mgmt-att-field ":" key-mgmt-att-value

key-mgmt-att-field = "key-mgmt"
key-mgmt-att-value = 0*1SP prtcl-id SP keymgmt-data

prtcl-id     = KMPID
              ; e.g. "mikey"

keymgmt-data = base64
SP           = 0x20
```

Location

SdpParser/CSdpFieldAttributeKeyMgmt.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeKeyMgmt (↗see page 136)	Default construtor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (↗see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldAttributeKeyMgmt (↗see page 136)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (↗see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (↗see page 139)	Assignment operator.
◆ == (↗see page 140)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (↗see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆V GenerateCopy (↗see page 137)	Generates a copy of the object.
◆V GenerateParameter (↗see page 137)	Generates a key management parameter.
◆ GetKeyManagementProtocol (↗see page 137)	Gets the type of the key management protocol.
◆ GetKeyManagementRole (↗see page 137)	Gets the role where this key management attribute is used.
◆ GetProtocolId (↗see page 137)	Gets the protocol string identifier.
◆ GetValue (↗see page 138)	Gets the value of the key management attribute.
◆V Parse (↗see page 138)	Parses the parameters list beginning at rpszStartPosition.
◆V Reset (↗see page 138)	Resets this object.
◆V Serialize (↗see page 138)	Outputs the data members to a blob.
◆ SetKeyManagementRole (↗see page 139)	Sets the role where this key management attribute is used. Parameter: eRole: The key management role.

◆ SetProtocolId (see page 139)	Sets the protocol string identifier.
◆ SetValue (see page 139)	Sets the value of the key management attribute.
◆ Ⓥ Validate (see page 139)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ Ⓐ Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ Ⓥ Reset (see page 353)	Resets the data in the parser.
◆ Ⓐ Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
Ⓥ	virtual
Ⓐ	abstract

Enumerations

Enumeration	Description
EKeyManagementAttributeRole (see page 140)	Tells in which scenario the key management attribute is used.
EKeyManagementProtocol (see page 140)	Defines the supported types of key management protocols.

10.1.15.1 - Constructors

10.1.15.1.1 - CSdpFieldAttributeKeyMgmt

10.1.15.1.1.1 - CSdpFieldAttributeKeyMgmt::CSdpFieldAttributeKeyMgmt Constructor

Default construtor.

C++

```
CSdpFieldAttributeKeyMgmt();
```

Description

Constructor

10.1.15.1.1.2 - CSdpFieldAttributeKeyMgmt::CSdpFieldAttributeKeyMgmt Constructor

Copy constructor.

C++

```
CSdpFieldAttributeKeyMgmt(IN const CSdpFieldAttributeKeyMgmt& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmt& rFrom`	The object to be copied.

Description

Copy constructor

10.1.15.2 - Destructors

10.1.15.2.1 - CSdpFieldAttributeKeyMgmt::~CSdpFieldAttributeKeyMgmt Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeKeyMgmt();
```

Description

Destructor

10.1.15.3 - Methods

10.1.15.3.1 - CSdpFieldAttributeKeyMgmt::GenerateCopy Method

Generates a copy of the object.

C++

```
virtual GO CSdpFieldAttributeKeyMgmt* GenerateCopy() const;
```

Returns

A copy of this object. Ownership is given.

Description

Creates a copy of this object

10.1.15.3.2 - CSdpFieldAttributeKeyMgmt::GenerateParameter Method

Generates a key management parameter.

C++

```
virtual GO CSdpKeyManagementParameter* GenerateParameter() const;
```

Returns

A key management parameter for this object.

Description

Generates a CSdpKeyManagementParameter (see page 280) of the appropriate type for this key attribute.

10.1.15.3.3 - CSdpFieldAttributeKeyMgmt::GetKeyManagementProtocol Method

Gets the type of the key management protocol.

C++

```
EKeyManagementProtocol GetKeyManagementProtocol() const;
```

Returns

The type of key management protocol.

Description

Gets the type of key management protocol that this object represents.

10.1.15.3.4 - CSdpFieldAttributeKeyMgmt::GetKeyManagementRole Method

Gets the role where this key management attribute is used.

C++

```
EKeyManagementAttributeRole GetKeyManagementRole() const;
```

Returns

The key management role.

Description

Gets the key management role where the attribute will be effective.

10.1.15.3.5 - CSdpFieldAttributeKeyMgmt::GetProtocolId Method

Gets the protocol string identifier.

C++

```
const char* GetProtocolId() const;
```

Returns

The protocol ID.

Description

Returns the protocol ID.

10.1.15.3.6 - CSdpFieldAttributeKeyMgmt::GetValue Method

Gets the value of the key management attribute.

C++

```
const char* GetValue() const;
```

Returns

The char representation of the key management attribute.

Description

Gets char representation of the key management attribute.

10.1.15.3.7 - CSdpFieldAttributeKeyMgmt::Parse Method

Parses the parameters list beginning at rpszStartPosition.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rRes);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rRes`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rRes' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.15.3.8 - CSdpFieldAttributeKeyMgmt::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 138).

10.1.15.3.9 - CSdpFieldAttributeKeyMgmt::Serialize Method

Outputs the data members to a blob.

C++

```
virtual void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.15.3.10 - CSdpFieldAttributeKeyMgmt::SetKeyManagementRole Method

Sets the role where this key management attribute is used.

Parameter: eRole: The key management role.

C++

```
void SetKeyManagementRole(IN const EKeyManagementAttributeRole eRole);
```

Description

Sets the key management role where the attribute will be effective.

10.1.15.3.11 - CSdpFieldAttributeKeyMgmt::SetProtocolId Method

Sets the protocol string identifier.

C++

```
void SetProtocolId(IN const char* szFormat);
```

Description

Sets the protocol ID.

10.1.15.3.12 - CSdpFieldAttributeKeyMgmt::SetValue Method

Sets the value of the key management attribute.

C++

```
void SetValue(IN const char* szValue);
```

Description

Sets the value of the key.

10.1.15.3.13 - CSdpFieldAttributeKeyMgmt::Validate Method

Returns true if data members are valid.

C++

```
virtual bool Validate();
```

Returns

False if one of the data members is empty, true otherwise.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.15.4 - Operators

10.1.15.4.1 - CSdpFieldAttributeKeyMgmt::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeKeyMgmt& operator =(IN const CSdpFieldAttributeKeyMgmt& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmt& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.15.4.2 - CSdpFieldAttributeKeyMgmt::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeKeyMgmt& rFrom) const;
```

Returns

true if both attributes contain the same key management attribute.

Description

Comparison operator

10.1.15.5 - Enumerations

10.1.15.5.1 - CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole Enumeration

Tells in which scenario the key management attribute is used.

C++

```
enum EKeyManagementAttributeRole {
  eNONE,
  eOFFER,
  eANSWER,
  eBOTH
};
```

Description

Tells in which scenario the key management attribute is used.

Members

Members	Description
eNONE	The key management attribute is never used.
eOFFER	The key management attribute is used for offers only.
eANSWER	The key management attribute is used for answers only.
eBOTH	The key management attribute is used for offers and answers.

10.1.15.5.2 - CSdpFieldAttributeKeyMgmt::EKeyManagementProtocol Enumeration

Defines the supported types of key management protocols.

C++

```
enum EKeyManagementProtocol {
  eGENERIC,
  eMIKEY
};
```

Description

Defines the types of key management protocols that are supported by this object and by its child classes.

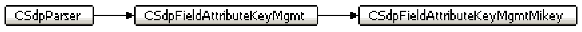
Members

Members	Description
eGENERIC	Generic describes all types of key management protocols that are different from MIKEY.
eMIKEY	The MIKEY key exchange protocol.

10.1.16 - CSdpFieldAttributeKeyMgmtMikey Class

This class implements an abstraction of a MIKEY key-mgmt-attribute.

Class Hierarchy



C++

```
class CSdpFieldAttributeKeyMgmtMikey : public CSdpFieldAttributeKeyMgmt;
```

Description

This class is an abstraction of a MIKEY key-mgmt-attribute in a SDP packet.

```
key-mgmt-attribute = key-mgmt-att-field ":" key-mgmt-att-value

key-mgmt-att-field = "key-mgmt"
key-mgmt-att-value = 0*1SP prtcl-id SP keymgmt-data

prtcl-id     = "mikey"

keymgmt-data = base64
SP           = 0x20
```

The attribute contains the Base64 encoded MIKEY message.

A CSdpFieldAttributeKeyMgmt (see page 134) class that returns a type eMIKEY from the GetKeyManagementProtocol (see page 137) method can be safely used as a CSdpFieldAttributeKeyMgmtMikey without any side effects.

Location

SdpParser/CSdpFieldAttributeKeyMgmtMikey.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeKeyMgmtMikey (see page 143)	Default constructor.

CSdpFieldAttributeKeyMgmt Class

CSdpFieldAttributeKeyMgmt Class	Description
◆ CSdpFieldAttributeKeyMgmt (see page 136)	Default construtor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆▽ ~CSdpFieldAttributeKeyMgmtMikey (see page 144)	Destructor.

CSdpFieldAttributeKeyMgmt Class

CSdpFieldAttributeKeyMgmt Class	Description
◆▽ ~CSdpFieldAttributeKeyMgmt (see page 136)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆▽ ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
▽	virtual

Operators

Operator	Description
◆ = (see page 148)	Assignment operator.

== (see page 149)	Comparison operator.

CSdpFieldAttributeKeyMgmt Class

CSdpFieldAttributeKeyMgmt Class	Description
= (see page 139)	Assignment operator.
== (see page 140)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
= (see page 354)	Assignment operator.

Legend

	Method

Methods

Method	Description
GenerateCopy (see page 144)	Generates a copy of the object.
GenerateParameter (see page 144)	Generates key management parameter.
GetErrorData (see page 144)	Gets the error data from a received MIKEY error message.
GetMikey (see page 144)	Gets the stored IMikey interace.
GetMikeyCryptoSessionBundle (see page 145)	Gets the crypto session bundle stored.
GetMikeyCryptoSessionBundleId (see page 145)	Gives the crypto session bundle ID.
GetSdpKeyMgmtIds (see page 145)	Gets the list of SDP key mgmt IDs from the general extension data from the attribute.
IsResponse (see page 146)	Checks if the MIKEY key management is currently sending a response.
ParseMikey (see page 146)	Parses the MIKEY message. Parameter: presMikeyError: Error returned by a parsed MIKEY error message.
ParseMikeyMessage (see page 146)	Parses the CMikeyMessage from the value stored in the member.
Reset (see page 146)	Resets this object.
Serialize (see page 147)	Serializes the MIKEY message.
SetErrorData (see page 147)	Sets the error data to send a MIKEY error message.
SetMikey (see page 147)	Sets the stored IMikey interace.
SetMikeyCryptoSessionBundle (see page 148)	Sets the crypto session bundle for the attribute.
SetSdpKeyMgmtIds (see page 148)	Sets the general extension data for the attribute.
ValidateSdpKeyMgmtIds (see page 148)	Validates that the General Extension is valid.

CSdpFieldAttributeKeyMgmt Class

CSdpFieldAttributeKeyMgmt Class	Description
GenerateCopy (see page 137)	Generates a copy of the object.
GenerateParameter (see page 137)	Generates a key management parameter.
GetKeyManagementProtocol (see page 137)	Gets the type of the key management protocol.
GetKeyManagementRole (see page 137)	Gets the role where this key management attribute is used.
GetProtocolId (see page 137)	Gets the protocol string identifier.
GetValue (see page 138)	Gets the value of the key management attribute.
Parse (see page 138)	Parses the parameters list beginning at rpszStartPosition.
Reset (see page 138)	Resets this object.
Serialize (see page 138)	Outputs the data members to a blob.
SetKeyManagementRole (see page 139)	Sets the role where this key management attribute is used. Parameter: eRole: The key management role.
SetProtocolId (see page 139)	Sets the protocol string identifier.
SetValue (see page 139)	Sets the value of the key management attribute.
Validate (see page 139)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
Reset (see page 353)	Resets the data in the parser.
Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

Enumerations

CSdpFieldAttributeKeyMgmt Class

CSdpFieldAttributeKeyMgmt Class	Description
EKeyManagementAttributeRole (see page 140)	Tells in which scenario the key management attribute is used.
EKeyManagementProtocol (see page 140)	Defines the supported types of key management protocols.

10.1.16.1 - Constructors

10.1.16.1.1 - CSdpFieldAttributeKeyMgmtMikey

10.1.16.1.1.1 - CSdpFieldAttributeKeyMgmtMikey::CSdpFieldAttributeKeyMgmtMikey Constructor

Default constructor.

C++

```
CSdpFieldAttributeKeyMgmtMikey();
```

Description

Constructor.

10.1.16.1.1.2 - CSdpFieldAttributeKeyMgmtMikey::CSdpFieldAttributeKeyMgmtMikey Constructor

Constructor.

C++

```
CSdpFieldAttributeKeyMgmtMikey(IN const CSdpFieldAttributeKeyMgmt& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmt& rFrom`	The CSdpFieldAttributeKeyMgmt (see page 134) to be copied.

Description

Copy constructor.

10.1.16.1.1.3 - CSdpFieldAttributeKeyMgmtMikey::CSdpFieldAttributeKeyMgmtMikey Constructor

Copy constructor.

C++

```
CSdpFieldAttributeKeyMgmtMikey(IN const CSdpFieldAttributeKeyMgmtMikey& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmtMikey& rFrom`	The CSdpFieldAttributeKeyMgmtMikey to be copied.

Description

Copy constructor.

10.1.16.2 - Destructors

10.1.16.2.1 - CSdpFieldAttributeKeyMgmtMikey::~CSdpFieldAttributeKeyMgmtMikey Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeKeyMgmtMikey();
```

Description

Destructor.

10.1.16.3 - Methods

10.1.16.3.1 - CSdpFieldAttributeKeyMgmtMikey::GenerateCopy Method

Generates a copy of the object.

C++

```
virtual GO CSdpFieldAttributeKeyMgmt* GenerateCopy() const;
```

Returns

A copy of this object. Ownership is given.

Description

Creates a copy of this object

10.1.16.3.2 - CSdpFieldAttributeKeyMgmtMikey::GenerateParameter Method

Generates key management parameter.

C++

```
virtual GO CSdpKeyManagementParameter* GenerateParameter() const;
```

Returns

A key management parameter for this object.

Description

Creates a CSdpKeyManagementParameter (see page 280) of the appropriate type for this key attribute.

10.1.16.3.3 - CSdpFieldAttributeKeyMgmtMikey::GetErrorData Method

Gets the error data from a received MIKEY error message.

C++

```
mxt_result GetErrorData(OUT mxt_result* presMikeyError);
```

Parameters

Parameters	Description
`OUT mxt_result* presMikeyError`	The first error reported by a MIKEY error message.

Returns

resS_OK: error code returned. resFE_INVALID_ARGUMENT: presMikeyError pointer is NULL.

Description

Gets the error returned by the contents of a MIKEY error message. If resS_OK is returned in the error, then no error message was received.

10.1.16.3.4 - CSdpFieldAttributeKeyMgmtMikey::GetMikey Method

Gets the stored IMikey interace.

C++

```
mxt_result GetMikey(OUT IMikey** ppMikey);
```

Parameters

Parameters	Description
`OUT IMikey** ppMikey`	The IMikey interface to use with this attribute.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: Pointer is NULL -resFE_INVALID_STATE: no IMikey set.

Description

Gets the IMikey interface used by this attribute.

10.1.16.3.5 - CSdpFieldAttributeKeyMgmtMikey::GetMikeyCryptoSessionBundle Method

Gets the crypto session bundle stored.

C++

```
mxt_result GetMikeyCryptoSessionBundle(OUT IMikeyCryptoSessionBundle** ppCryptoSessionBundle);
```

Parameters

Parameters	Description
`OUT IMikeyCryptoSessionBundle** ppCryptoSessionBundle`	The IMikeyCryptoSessionBundle interface contained in this attribute.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: ppCryptoSessionBundle is NULL. -resFE_INVALID_STATE: crypto session bundle is not set.

Description

Gets the IMikeyCryptoSessionBundle interface contained in this attribute.

NOTES: It is up to the application to release the interface when it is finished using it.

10.1.16.3.6 - CSdpFieldAttributeKeyMgmtMikey::GetMikeyCryptoSessionBundleId Method

Gives the crypto session bundle ID.

C++

```
mxt_result GetMikeyCryptoSessionBundleId(INOUT uint32_t* puId);
```

Parameters

Parameters	Description
`INOUT uint32_t* puId`	Pointer to an uint32_t where the crypto session bundle ID is set.

Returns

resFE_INVALID_STATE: If the crypto session bundle is NULL. Value return by IMikeyCryptoSessionBundle::GetCryptoSessionBundleIdentifier.

Description

Returns the crypto session bundle ID contained in the MIKEY message. ParseMikeyMessage (see page 146) must have been previously called in order to access the MIKEY message.

See Also

ParseMikeyMessage (see page 146)

10.1.16.3.7 - CSdpFieldAttributeKeyMgmtMikey::GetSdpKeyMgmtIds Method

Gets the list of SDP key mgmt IDs from the general extension data from the attribute.

C++

```
mxt_result GetSdpKeyMgmtIds(OUT CBlob* pData);
```

Parameters

Parameters	Description
`OUT CBlob* pData`	The SDP key management IDs to retrieve from the general extension.

Returns

resS_OK: General extension SDP IDs data properly extracted. resFE_INVALID_ARGUMENT: pData pointer is NULL.

Description

Gets the SDP key management IDs from the general extension data from this attribute.

10.1.16.3.8 - CSdpFieldAttributeKeyMgmtMikey::IsResponse Method

Checks if the MIKEY key management is currently sending a response.

C++

```
bool IsResponse();
```

Returns

-true: This is a response to an initiation. -false: Not answering an initiation.

Description

Validates if this attribute is currently capable of sending a response.

10.1.16.3.9 - CSdpFieldAttributeKeyMgmtMikey::ParseMikey Method

Parses the MIKEY message.

Parameter: presMikeyError: Error returned by a parsed MIKEY error message.

C++

```
mxt_result ParseMikey(OUT mxt_result * presMikeyError);
```

Returns

resFE_INVALID_STATE: no crypto session bundle exists or no IMikey interface set. resFE_FAIL: The crypto session bundle ID in the message does not match the one on the attribute

Description

Parses the MIKEY message. The message is handled by the MIEKY stack and this method has no influence on the SDP parsing.

10.1.16.3.10 - CSdpFieldAttributeKeyMgmtMikey::ParseMikeyMessage Method

Parses the CMikeyMessage from the value stored in the member.

C++

```
mxt_result ParseMikeyMessage();
```

Returns

Value returned by: CreateInternalMessage() CBase64:Begin() CBase64:Update() CBase64:End() IMikeyMessage::Parse (see page 138)()

Description

This method parses the value contained in the attribute into an IMikeyMessage.

10.1.16.3.11 - CSdpFieldAttributeKeyMgmtMikey::Reset Method

Resets this object.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 138).

10.1.16.3.12 - Serialize

10.1.16.3.12.1 - CSdpFieldAttributeKeyMgmtMikey::Serialize Method

Serializes the MIKEY message.

C++

```
void Serialize();
```

Description

Serializes the MIKEY message and fills up the base class value.

10.1.16.3.12.2 - CSdpFieldAttributeKeyMgmtMikey::Serialize Method

Outputs the data members to a blob.

C++

```
virtual void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.16.3.13 - CSdpFieldAttributeKeyMgmtMikey::SetErrorData Method

Sets the error data to send a MIKEY error message.

C++

```
mxt_result SetErrorData(IN mxt_result resMikeyError);
```

Parameters

Parameters	Description
`IN mxt_result resMikeyError`	The error to send to the peer via MIKEY.

Returns

resS_OK: Error set. resFE_INVALID_ARGUMENT: Error is not a MIKEY error defined in the MIKEY implementation.

Description

Sets MIKEY error data to generate an error message.

10.1.16.3.14 - CSdpFieldAttributeKeyMgmtMikey::SetMikey Method

Sets the stored IMikey interace.

C++

```
mxt_result SetMikey(IN IMikey* pMikey);
```

Parameters

Parameters	Description
`IN IMikey* pMikey`	The IMikey interface to use with this attribute.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: Pointer is NULL.

Description

Sets the IMikey interface that this attribute uses.

10.1.16.3.15 - CSdpFieldAttributeKeyMgmtMikey::SetMikeyCryptoSessionBundle Method

Sets the crypto session bundle for the attribute.

C++

```
mxt_result SetMikeyCryptoSessionBundle(IN IMikeyCryptoSessionBundle* pCryptoSessionBundle);
```

Parameters

Parameters	Description
`IN IMikeyCryptoSessionBundle* pCryptoSessionBundle`	The IMikeyCryptoSessionBundle interface to use with this attribute.

Returns

-resS_OK: Operation successful. -resFE_INVALID_STATE: No IMikey set. -resFE_INVALID_ARGUMENT: NULL pointer.

Description

Sets the IMikeyCryptoSessionBundle interface to use with this attribute.

10.1.16.3.16 - CSdpFieldAttributeKeyMgmtMikey::SetSdpKeyMgmtIds Method

Sets the general extension data for the attribute.

C++

```
mxt_result SetSdpKeyMgmtIds(IN const CBlob* pData);
```

Parameters

Parameters	Description
`IN const CBlob* pData`	The SDP key management IDs data to set.

Returns

resS_OK: SDP key management IDs data properly set. resFE_INVALID_ARGUMENT: NULL pointer.

Description

Sets the SDP key management IDs data.

10.1.16.3.17 - CSdpFieldAttributeKeyMgmtMikey::ValidateSdpKeyMgmtIds Method

Validates that the General Extension is valid.

C++

```
mxt_result ValidateSdpKeyMgmtIds(IN CBlob* pData);
```

Parameters

Parameters	Description
`IN CBlob* pData`	The general extension data to validate.

Returns

resS_OK: The data is valid. resFE_FAIL: Data is invalid. resFE_INVALID_ARGUMENT: NULL pointer.

Description

Validates the general extension SDP key management IDs data to see if it corresponds to what is expected.

10.1.16.4 - Operators

10.1.16.4.1 - CSdpFieldAttributeKeyMgmtMikey::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeKeyMgmtMikey& operator =(IN const CSdpFieldAttributeKeyMgmtMikey& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmtMikey& rFrom`	The CSdpFieldAttributeKeyMgmtMikey (see page 140) to be copied.

Description

Assignment operator.

10.1.16.4.2 - CSdpFieldAttributeKeyMgmtMikey::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeKeyMgmtMikey& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmtMikey& rFrom`	The CSdpFieldAttributeKeyMgmtMikey (see page 140) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.17 - CSdpFieldAttributeMaxBitRate Class

This class implements an abstraction of an attribute-max-bit-rate.

Class Hierarchy

C++

```
class CSdpFieldAttributeMaxBitRate : public CSdpParser;
```

Description

This class is an abstraction of an attribute-max-bit-rate in a SDP packet.

The parsing of this attribute-max-bit-rate is a specific case of an attribute. The basic BNF that an attribute can have is described into CSdpFieldAttributeOther (see page 160).

```
attribute-max-bit-rate  =   "T38MaxBitRate:" byte-string
byte-string             =   1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeMaxBitRate.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeMaxBitRate (see page 150)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeMaxBitRate (see page 151)	Destructor

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 152)	Assignment operator.
◆ == (see page 153)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetMaxBitRate (see page 151)	Gets the maximum bitrate value.
◆ Parse (see page 151)	Parses all the needed information for this field.
◆ Reset (see page 151)	Resets all member variables.
◆ Serialize (see page 152)	Generates the data blob from the data members.
◆ SetMaxBitRate (see page 152)	Sets the maximum bitrate value.
◆ Validate (see page 152)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.17.1 - Constructors

10.1.17.1.1 - CSdpFieldAttributeMaxBitRate

10.1.17.1.1.1 - CSdpFieldAttributeMaxBitRate::CSdpFieldAttributeMaxBitRate Constructor

Default constructor.

C++

```
CSdpFieldAttributeMaxBitRate();
```

Description

Constructor

10.1.17.1.1.2 - CSdpFieldAttributeMaxBitRate::CSdpFieldAttributeMaxBitRate Constructor

Copy constructor.

C++

```
CSdpFieldAttributeMaxBitRate(IN const CSdpFieldAttributeMaxBitRate& rFrom);
```

Description

Copy constructor

10.1.17.2 - Destructors

10.1.17.2.1 - CSdpFieldAttributeMaxBitRate::~CSdpFieldAttributeMaxBitRate Destructor

Destructor

C++

```
virtual ~CSdpFieldAttributeMaxBitRate();
```

Description

Destructor

10.1.17.3 - Methods

10.1.17.3.1 - CSdpFieldAttributeMaxBitRate::GetMaxBitRate Method

Gets the maximum bitrate value.

C++

```
const uint32_t GetMaxBitRate() const;
```

Returns

The maximum bitrate.

Description

Returns the maximum bitrate.

10.1.17.3.2 - CSdpFieldAttributeMaxBitRate::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.17.3.3 - CSdpFieldAttributeMaxBitRate::Reset Method

Resets all member variables.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 151).

10.1.17.3.4 - CSdpFieldAttributeMaxBitRate::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.17.3.5 - CSdpFieldAttributeMaxBitRate::SetMaxBitRate Method

Sets the maximum bitrate value.

C++

```
void SetMaxBitRate(IN const uint32_t nMaxBitRate);
```

Parameters

Parameters	Description
`IN const uint32_t nMaxBitRate`	The maximum bitrate to set.

Description

Sets the maximum bitrate.

10.1.17.3.6 - CSdpFieldAttributeMaxBitRate::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.17.4 - Operators

10.1.17.4.1 - CSdpFieldAttributeMaxBitRate::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeMaxBitRate& operator =(IN const CSdpFieldAttributeMaxBitRate& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMaxBitRate& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.17.4.2 - CSdpFieldAttributeMaxBitRate::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeMaxBitRate& rFrom) const;
```

Returns

true if both attributes contain the same bitrate.

Description

Comparison operator

10.1.18 - CSdpFieldAttributeMaxDatagram Class

This class implements abstraction of an attribute-max-datagram.

Class Hierarchy

CSdpParser
CSdpFieldAttributeMaxDatagram

C++

```
class CSdpFieldAttributeMaxDatagram : public CSdpParser;
```

Description

This class is an abstraction of an attribute-max-datagram in a SDP packet.

The parsing of this attribute-max-datagrame is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-max-datagram  =   "T38FaxMaxDatagram:" byte-string
byte-string             =   1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeMaxDatagram.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeMaxDatagram (see page 154)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldAttributeMaxDatagram (see page 155)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 156)	Assignment operator.
◆ == (see page 156)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetMaxDatagram (see page 155)	Gets the attribute max datagram.
◆ Parse (see page 155)	Parses all the needed information for this field.
◆ Reset (see page 155)	Resets all the data members.
◆ Serialize (see page 155)	Generates the data blob from the data members.
◆ SetMaxDatagram (see page 156)	Sets the attribute max datagram.
◆ Validate (see page 156)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.18.1 - Constructors

10.1.18.1.1 - CSdpFieldAttributeMaxDatagram

10.1.18.1.1.1 - CSdpFieldAttributeMaxDatagram::CSdpFieldAttributeMaxDatagram Constructor

Default constructor.

C++

```
CSdpFieldAttributeMaxDatagram();
```

Description

Constructor

10.1.18.1.1.2 - CSdpFieldAttributeMaxDatagram::CSdpFieldAttributeMaxDatagram Constructor

Copy constructor.

C++

```
CSdpFieldAttributeMaxDatagram(IN const CSdpFieldAttributeMaxDatagram& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMaxDatagram& rFrom`	The object to be copied.

Description

Copy constructor

10.1.18.2 - Destructors

10.1.18.2.1 - CSdpFieldAttributeMaxDatagram::~CSdpFieldAttributeMaxDatagram Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeMaxDatagram();
```

Description

Destructor

10.1.18.3 - Methods

10.1.18.3.1 - CSdpFieldAttributeMaxDatagram::GetMaxDatagram Method

Gets the attribute max datagram.

C++

```
const int32_t GetMaxDatagram() const;
```

Returns

The attribute max-datagram.

Description

Returns the max-datagram.

10.1.18.3.2 - CSdpFieldAttributeMaxDatagram::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.18.3.3 - CSdpFieldAttributeMaxDatagram::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 155).

10.1.18.3.4 - CSdpFieldAttributeMaxDatagram::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.18.3.5 - CSdpFieldAttributeMaxDatagram::SetMaxDatagram Method

Sets the attribute max datagram.

C++

```
void SetMaxDatagram(IN const int nMaxDatagram);
```

Parameters

Parameters	Description
`IN const int nMaxDatagram`	The value of the attribute max-datagram to set.

Description

Sets the attribute max-datagram.

10.1.18.3.6 - CSdpFieldAttributeMaxDatagram::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.18.4 - Operators

10.1.18.4.1 - CSdpFieldAttributeMaxDatagram::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeMaxDatagram& operator =(IN const CSdpFieldAttributeMaxDatagram& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMaxDatagram& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.18.4.2 - CSdpFieldAttributeMaxDatagram::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeMaxDatagram& rFrom) const;
```

Returns

true if both attributes contain the same max datagram value.

Description

Comparison operator

10.1.19 - CSdpFieldAttributeMid Class

This class implements an abstraction of the mid attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeMid

C++

```
class CSdpFieldAttributeMid : public CSdpParser;
```

Description

This class is an abstraction of the mid attribute in SDP. The mid field attribute is used to identify a media stream within a session description. It follows the BNF notation described in RFC 3388.

```
mid-attribute      = "a=mid:" identification-tag
identification-tag = token
```

Location

SdpParser/CSdpFieldAttributeMid.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeMid (see page 158)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldAttributeMid (see page 158)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 160)	Assignment Operator.
◆ == (see page 160)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetValue (see page 159)	Gets the value.
◆ V Parse (see page 159)	Parses the data.
◆ V Reset (see page 159)	Resets the data in the parser.
◆ Serialize (see page 159)	Serializes the value into the blob.
◆ SetValue (see page 159)	Sets the value.
◆ V Validate (see page 160)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.19.1 - Constructors

10.1.19.1.1 - CSdpFieldAttributeMid

10.1.19.1.1.1 - CSdpFieldAttributeMid::CSdpFieldAttributeMid Constructor

Default constructor.

C++

```
CSdpFieldAttributeMid();
```

Description

Constructor

10.1.19.1.1.2 - CSdpFieldAttributeMid::CSdpFieldAttributeMid Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeMid(IN const CSdpFieldAttributeMid& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMid& rSrc`	The CSdpFieldAttributeMid to be copied.

Description

Copy constructor

10.1.19.2 - Destructors

10.1.19.2.1 - CSdpFieldAttributeMid::~CSdpFieldAttributeMid Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeMid();
```

Description

Destructor

10.1.19.3 - Methods

10.1.19.3.1 - CSdpFieldAttributeMid::GetValue Method

Gets the value.

C++

```
const char* GetValue() const;
```

Returns

The mid value.

Description

Returns the mid value.

10.1.19.3.2 - CSdpFieldAttributeMid::Parse Method

Parses the data.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

10.1.19.3.3 - CSdpFieldAttributeMid::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 159).

10.1.19.3.4 - CSdpFieldAttributeMid::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.19.3.5 - CSdpFieldAttributeMid::SetValue Method

Sets the value.

C++

```
void SetValue(IN const char* szValue);
```

Parameters

Parameters	Description
`IN const char* szValue`	The mid value to set.

Description

Sets the mid value.

10.1.19.3.6 - CSdpFieldAttributeMid::Validate Method

Validates the parsed data.

C++

```
virtual bool Validate();
```

Returns

False if one of the data members is empty, true otherwise.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.19.4 - Operators

10.1.19.4.1 - CSdpFieldAttributeMid::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeMid& operator =(IN const CSdpFieldAttributeMid& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMid& rSrc`	The CSdpFieldAttributeMid (⧉see page 157) to be copied.

Description

Assignment operator.

10.1.19.4.2 - CSdpFieldAttributeMid::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeMid& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeMid& rFrom`	The CSdpFieldAttributeKeyMgmtMikey (⧉see page 140) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.20 - CSdpFieldAttributeOther Class

This class implements an abstraction of an attribute-other.

Class Hierarchy

CSdpParser
CSdpFieldAttributeOther

C++

```
class CSdpFieldAttributeOther : public CSdpParser;
```

Description

This class is an abstraction of an attribute-other (other than those that are recognized) in a SDP packet. It follows the BNF notation described in RFC 2327.

RFC 2327 BNF:

```
attribute-other   =    (att-field ":" att-value) | att-field
att-value         =    byte-string
att-field         =    1*(alpha-numeric)
byte-string       =    1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeOther.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeOther (see page 162)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeOther (see page 162)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 164)	Assignment operator.
◆ == (see page 164)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetName (see page 162)	Gets the name.
◆ GetValue (see page 163)	Gets the value.
◆ Parse (see page 163)	Parses all the needed information for this field.
◆ Reset (see page 163)	Resets the data in the parser.
◆ Serialize (see page 163)	Generates the data blob from the data members.
◆ SetName (see page 164)	Sets the name.
◆ SetValue (see page 164)	Sets the value.
◆ Validate (see page 164)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.

◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.20.1 - Constructors

10.1.20.1.1 - CSdpFieldAttributeOther

10.1.20.1.1.1 - CSdpFieldAttributeOther::CSdpFieldAttributeOther Constructor

Default constructor.

C++

```
CSdpFieldAttributeOther();
```

Description

Constructor

10.1.20.1.1.2 - CSdpFieldAttributeOther::CSdpFieldAttributeOther Constructor

Copy constructor.

C++

```
CSdpFieldAttributeOther(IN const CSdpFieldAttributeOther& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeOther& rFrom`	The object to be copied.

Description

Copy constructor

10.1.20.2 - Destructors

10.1.20.2.1 - CSdpFieldAttributeOther::~CSdpFieldAttributeOther Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeOther();
```

Description

Destructor

10.1.20.3 - Methods

10.1.20.3.1 - CSdpFieldAttributeOther::GetName Method

Gets the name.

C++

```
const char* GetName() const;
```

Returns

The name of the attribute-other.

Description

Gets the name of the attribute-other

10.1.20.3.2 - CSdpFieldAttributeOther::GetValue Method

Gets the value.

C++

```
const char* GetValue() const;
```

Returns

The value of the attribute-other.

Description

Gets the value of the attribute-other

10.1.20.3.3 - CSdpFieldAttributeOther::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.20.3.4 - CSdpFieldAttributeOther::Reset Method

Resets the data in the parser.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 163).

10.1.20.3.5 - CSdpFieldAttributeOther::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.20.3.6 - CSdpFieldAttributeOther::SetName Method

Sets the name.

C++

```
void SetName(IN const char* pszName);
```

Parameters

Parameters	Description
`IN const char* pszName`	The name of the attribute-other.

Description

Sets the name of the attribute-other

10.1.20.3.7 - CSdpFieldAttributeOther::SetValue Method

Sets the value.

C++

```
void SetValue(IN const char* pszValue);
```

Parameters

Parameters	Description
`IN const char* pszValue`	The value of the attribute-other.

Description

Sets the value of the attribute-other

10.1.20.3.8 - CSdpFieldAttributeOther::Validate Method

Validates the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.20.4 - Operators

10.1.20.4.1 - CSdpFieldAttributeOther::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeOther& operator =(IN const CSdpFieldAttributeOther& rFrom);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.20.4.2 - CSdpFieldAttributeOther::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeOther& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.21 - CSdpFieldAttributePreCond Class

This class implements an abstraction of the precondition field attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributePreCond

C++

```
class CSdpFieldAttributePreCond : public CSdpParser;
```

Description

This class is an abstraction of the precondition field attribute in SDP from RFC 3312.

```
desired-status     =  "a=des:" precondition-type
                       SP strength-tag SP status-type
                       SP direction-tag
current-status     =  "a=curr:" precondition-type
                      SP status-type SP direction-tag
confirm-status     =  "a=conf:" precondition-type
                      SP status-type SP direction-tag
precondition-type  =  "qos" | token
strength-tag       =  ("mandatory" | "optional" | "none"
                   =  | "failure" | "unknown")
status-type        =  ("e2e" | "local" | "remote")
direction-tag      =  ("none" | "send" | "recv" | "sendrecv")
```

Location

SdpParser/CSdpFieldAttributePreCond.h

Constructors

Constructor	Description
◆ CSdpFieldAttributePreCond (see page 166)	Copy Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributePreCond (see page 166)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 169)	Assignment Operator.
◆ == (see page 169)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetDirectionTag (see page 167)	Gets the Direction tag.
◆ GetPrecondType (see page 167)	Gets the Precondition type.
◆ GetStatusTag (see page 167)	Gets the Status tag.
◆ V Parse (see page 167)	Parses the data.
◆ V Reset (see page 167)	Resets the data in the parser.
◆ Serialize (see page 168)	Serializes the value into the blob.
◆ SetDirectionTag (see page 168)	Sets the Direction tag.
◆ SetPrecondType (see page 168)	Sets the Precondition type.
◆ SetStatusTag (see page 168)	Sets the Status tag.
◆ V Validate (see page 169)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.21.1 - Constructors

10.1.21.1.1 - CSdpFieldAttributePreCond::CSdpFieldAttributePreCond Constructor

Copy Constructor.

C++

```
CSdpFieldAttributePreCond(IN const CSdpFieldAttributePreCond& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCond& rSrc`	The CSdpFieldAttributePreCond to be copied.

Description

Copy constructor

10.1.21.2 - Destructors

10.1.21.2.1 - CSdpFieldAttributePreCond::~CSdpFieldAttributePreCond Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributePreCond();
```

Description

Destructor

10.1.21.3 - Methods

10.1.21.3.1 - CSdpFieldAttributePreCond::GetDirectionTag Method

Gets the Direction tag.

C++

```
const CSdpParser::EPreCondDirectionTag GetDirectionTag() const;
```

Returns

The direction tag.

Description

Returns the direction tag.

10.1.21.3.2 - CSdpFieldAttributePreCond::GetPrecondType Method

Gets the Precondition type.

C++

```
const char* GetPrecondType() const;
```

Returns

A string that contains the precondition type.

Description

Returns a string that contains the precondition type.

10.1.21.3.3 - CSdpFieldAttributePreCond::GetStatusTag Method

Gets the Status tag.

C++

```
const CSdpParser::EPreCondStatusTag GetStatusTag() const;
```

Returns

The status parameter.

Description

Returns the status parameter.

10.1.21.3.4 - CSdpFieldAttributePreCond::Parse Method

Parses the data.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

10.1.21.3.5 - CSdpFieldAttributePreCond::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 167).

10.1.21.3.6 - CSdpFieldAttributePreCond::Serialize Method

Serializes the value into the blob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.21.3.7 - CSdpFieldAttributePreCond::SetDirectionTag Method

Sets the Direction tag.

C++

```
void SetDirectionTag(IN CSdpParser::EPreCondDirectionTag ePreCondDirectionTag);
```

Parameters

Parameters	Description
`IN CSdpParser::EPreCondDirectionTag ePreCondDirectionTag`	The precondition direction tag.

Description

Sets the precondition direction tag.

10.1.21.3.8 - CSdpFieldAttributePreCond::SetPrecondType Method

Sets the Precondition type.

C++

```
void SetPrecondType(IN const char* pszValue);
```

Parameters

Parameters	Description
`IN const char* pszValue`	String that contains the precondition type.

Description

Sets the string that contains the precondition type.

10.1.21.3.9 - CSdpFieldAttributePreCond::SetStatusTag Method

Sets the Status tag.

C++

```
void SetStatusTag(IN CSdpParser::EPreCondStatusTag ePreCondStatusTag);
```

Parameters

Parameters	Description
`IN CSdpParser::EPreCondStatusTag ePreCondStatusTag`	The precondition status parameter.

Description

Sets the precondition status parameter.

10.1.21.3.10 - CSdpFieldAttributePreCond::Validate Method

Validates the parsed data.

C++

```
virtual bool Validate();
```

Returns

False if one of the data members is empty or invalid, true otherwise.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.21.4 - Operators

10.1.21.4.1 - CSdpFieldAttributePreCond::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributePreCond& operator =(IN const CSdpFieldAttributePreCond& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCond& rSrc`	The CSdpFieldAttributePreCond (see page 165) to be copied.

Description

Assignment operator.

10.1.21.4.2 - CSdpFieldAttributePreCond::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributePreCond& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCond& rFrom`	The CSdpFieldAttributePreCond (see page 165) to be compared.

Returns

true if both are equal, false otherwise.

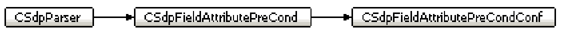
Description

Comparison operator.

10.1.22 - CSdpFieldAttributePreCondConf Class

This class implements a CONF precondition field attribute.

Class Hierarchy



C++

```
class CSdpFieldAttributePreCondConf : public CSdpFieldAttributePreCond;
```

Description

This class represents a CONF precondition field attribute.

Location

SdpParser/CSdpFieldAttributePreCondConf.h

Constructors

Constructor	Description
◆ CSdpFieldAttributePreCondConf (see page 171)	Default Constructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ CSdpFieldAttributePreCond (see page 166)	Copy Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributePreCondConf (see page 171)	Destructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ V ~CSdpFieldAttributePreCond (see page 166)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 171)	Assignment Operator.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ = (see page 169)	Assignment Operator.
◆ == (see page 169)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ GetDirectionTag (see page 167)	Gets the Direction tag.
◆ GetPrecondType (see page 167)	Gets the Precondition type.
◆ GetStatusTag (see page 167)	Gets the Status tag.
◆ V Parse (see page 167)	Parses the data.
◆ V Reset (see page 167)	Resets the data in the parser.
◆ Serialize (see page 168)	Serializes the value into the blob.
◆ SetDirectionTag (see page 168)	Sets the Direction tag.
◆ SetPrecondType (see page 168)	Sets the Precondition type.
◆ SetStatusTag (see page 168)	Sets the Status tag.

◆ V Validate (see page 169)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.22.1 - Constructors

10.1.22.1.1 - CSdpFieldAttributePreCondConf

10.1.22.1.1.1 - CSdpFieldAttributePreCondConf::CSdpFieldAttributePreCondConf Constructor

Default Constructor.

C++

```
CSdpFieldAttributePreCondConf();
```

Description

Constructor.

10.1.22.1.1.2 - CSdpFieldAttributePreCondConf::CSdpFieldAttributePreCondConf Constructor

Copy Constructor.

C++

```
CSdpFieldAttributePreCondConf(IN const CSdpFieldAttributePreCondConf& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondConf& rSrc`	The CSdpFieldAttributePreCond (see page 165) to be copied.

Description

Copy constructor

10.1.22.2 - Destructors

10.1.22.2.1 - CSdpFieldAttributePreCondConf::~CSdpFieldAttributePreCondConf Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributePreCondConf();
```

Description

Destructor

10.1.22.3 - Operators

10.1.22.3.1 - CSdpFieldAttributePreCondConf::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributePreCondConf& operator =(IN const CSdpFieldAttributePreCondConf& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondConf& rSrc`	The CSdpFieldAttributePreCond (see page 165) to be copied.

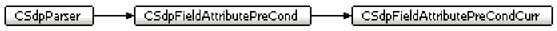
Description

Assignment operator.

10.1.23 - CSdpFieldAttributePreCondCurr Class

This class implements a CURR precondition field attribute.

Class Hierarchy



C++

```
class CSdpFieldAttributePreCondCurr : public CSdpFieldAttributePreCond;
```

Description

This class represents a CURR precondition field attribute.

Location

SdpParser/CSdpFieldAttributePreCondCurr.h

Constructors

Constructor	Description
◆ CSdpFieldAttributePreCondCurr (see page 173)	Default Constructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ CSdpFieldAttributePreCond (see page 166)	Copy Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributePreCondCurr (see page 174)	Destructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ V ~CSdpFieldAttributePreCond (see page 166)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 174)	Assignment Operator.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ = (see page 169)	Assignment Operator.
◆ == (see page 169)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ GetDirectionTag (see page 167)	Gets the Direction tag.
◆ GetPrecondType (see page 167)	Gets the Precondition type.
◆ GetStatusTag (see page 167)	Gets the Status tag.
◆V Parse (see page 167)	Parses the data.
◆V Reset (see page 167)	Resets the data in the parser.
◆ Serialize (see page 168)	Serializes the value into the blob.
◆ SetDirectionTag (see page 168)	Sets the Direction tag.
◆ SetPrecondType (see page 168)	Sets the Precondition type.
◆ SetStatusTag (see page 168)	Sets the Status tag.
◆V Validate (see page 169)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.23.1 - Constructors

10.1.23.1.1 - CSdpFieldAttributePreCondCurr

10.1.23.1.1.1 - CSdpFieldAttributePreCondCurr::CSdpFieldAttributePreCondCurr Constructor

Default Constructor.

C++

```
CSdpFieldAttributePreCondCurr();
```

Description

Constructor.

10.1.23.1.1.2 - CSdpFieldAttributePreCondCurr::CSdpFieldAttributePreCondCurr Constructor

Copy Constructor.

C++

```
CSdpFieldAttributePreCondCurr(IN const CSdpFieldAttributePreCondCurr& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondCurr& rSrc`	The CSdpFieldAttributePreCond (see page 165) to be copied.

Description

Copy constructor

10.1.23.2 - Destructors

10.1.23.2.1 - CSdpFieldAttributePreCondCurr::~CSdpFieldAttributePreCondCurr Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributePreCondCurr();
```

Description

Destructor

10.1.23.3 - Operators

10.1.23.3.1 - CSdpFieldAttributePreCondCurr::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributePreCondCurr& operator =(IN const CSdpFieldAttributePreCondCurr& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondCurr& rSrc`	The CSdpFieldAttributePreCond (see page 165) to be copied.

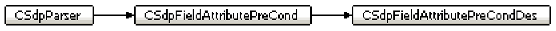
Description

Assignment operator.

10.1.24 - CSdpFieldAttributePreCondDes Class

This class implements a DES precondition field attribute.

Class Hierarchy



C++

```
class CSdpFieldAttributePreCondDes : public CSdpFieldAttributePreCond;
```

Description

This class represents a DES precondition field attribute.

Location

SdpParser/CSdpFieldAttributePreCondDes.h

Constructors

Constructor	Description
CSdpFieldAttributePreCondDes (see page 176)	Default Constructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
CSdpFieldAttributePreCond (see page 166)	Copy Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributePreCondDes (see page 176)	Destructor.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ V ~CSdpFieldAttributePreCond (see page 166)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 177)	Assignment Operator.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ = (see page 169)	Assignment Operator.
◆ == (see page 169)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetStrength (see page 176)	Gets the Strength tag.
◆ SetStrength (see page 177)	Sets the Strength tag.

CSdpFieldAttributePreCond Class

CSdpFieldAttributePreCond Class	Description
◆ GetDirectionTag (see page 167)	Gets the Direction tag.
◆ GetPrecondType (see page 167)	Gets the Precondition type.
◆ GetStatusTag (see page 167)	Gets the Status tag.
◆ V Parse (see page 167)	Parses the data.
◆ V Reset (see page 167)	Resets the data in the parser.
◆ Serialize (see page 168)	Serializes the value into the blob.
◆ SetDirectionTag (see page 168)	Sets the Direction tag.
◆ SetPrecondType (see page 168)	Sets the Precondition type.
◆ SetStatusTag (see page 168)	Sets the Status tag.
◆ V Validate (see page 169)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.

◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.24.1 - Constructors

10.1.24.1.1 - CSdpFieldAttributePreCondDes

10.1.24.1.1.1 - CSdpFieldAttributePreCondDes::CSdpFieldAttributePreCondDes Constructor

Default Constructor.

C++

```
CSdpFieldAttributePreCondDes();
```

Description

Constructor.

10.1.24.1.1.2 - CSdpFieldAttributePreCondDes::CSdpFieldAttributePreCondDes Constructor

Copy Constructor.

C++

```
CSdpFieldAttributePreCondDes(IN const CSdpFieldAttributePreCondDes& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondDes& rSrc`	The CSdpFieldAttributePreCondDes to be copied.

Description

Copy constructor

10.1.24.2 - Destructors

10.1.24.2.1 - CSdpFieldAttributePreCondDes::~CSdpFieldAttributePreCondDes Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributePreCondDes();
```

Description

Destructor

10.1.24.3 - Methods

10.1.24.3.1 - CSdpFieldAttributePreCondDes::GetStrength Method

Gets the Strength tag.

C++

```
const CSdpParser::EPreCondStrengthTag GetStrength() const;
```

Returns

The strength parameter.

Description

Returns the strength parameter.

10.1.24.3.2 - CSdpFieldAttributePreCondDes::SetStrength Method

Sets the Strength tag.

C++

```
void SetStrength(IN CSdpParser::EPreCondStrengthTag ePreCondStrength);
```

Parameters

Parameters	Description
`IN CSdpParser::EPreCondStrengthTag ePreCondStrength`	The precondition strength parameter.

Description

Sets the precondition strength parameter.

10.1.24.4 - Operators

10.1.24.4.1 - CSdpFieldAttributePreCondDes::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributePreCondDes& operator =(IN const CSdpFieldAttributePreCondDes& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePreCondDes& rSrc`	The CSdpFieldAttributePreCondDes (see page 174) to be copied.

Description

Assignment operator.

10.1.25 - CSdpFieldAttributePtime Class

This class implements an abstraction of an attribute-ptime.

Class Hierarchy

CSdpParser
CSdpFieldAttributePtime

C++

```
class CSdpFieldAttributePtime : public CSdpParser;
```

Description

This class is an abstraction of an attribute-ptime in a SDP packet.

The parsing of this attribute-ptime is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-ptime    =   "ptime:" byte-string
byte-string        =   1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributePtime.h

Constructors

Constructor	Description
CSdpFieldAttributePtime (see page 178)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributePtime (⧉see page 179)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (⧉see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (⧉see page 180)	Assignment Operator.
◆ == (⧉see page 181)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (⧉see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetPacketTime (⧉see page 179)	Gets the packet time.
◆ Parse (⧉see page 179)	Parses all the needed information for this field.
◆ Reset (⧉see page 180)	Resets this object.
◆ Serialize (⧉see page 180)	Generates the data blob from the data members.
◆ SetPacketTime (⧉see page 180)	Sets the packet time.
◆ Validate (⧉see page 180)	Returns true if data members are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (⧉see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (⧉see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (⧉see page 353)	Resets the data in the parser.
◆ A Validate (⧉see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.25.1 - Constructors

10.1.25.1.1 - CSdpFieldAttributePtime

10.1.25.1.1.1 - CSdpFieldAttributePtime::CSdpFieldAttributePtime Constructor

Default constructor.

C++

```
CSdpFieldAttributePtime();
```

Description

Constructor

10.1.25.1.1.2 - CSdpFieldAttributePtime::CSdpFieldAttributePtime Constructor

Copy constructor.

C++

```
CSdpFieldAttributePtime(IN const CSdpFieldAttributePtime& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePtime& rFrom`	The object to be copied.

Description

Copy constructor

10.1.25.2 - Destructors

10.1.25.2.1 - CSdpFieldAttributePtime::~CSdpFieldAttributePtime Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributePtime();
```

Description

Destructor

10.1.25.3 - Methods

10.1.25.3.1 - CSdpFieldAttributePtime::GetPacketTime Method

Gets the packet time.

C++

```
int32_t GetPacketTime() const;
```

Returns

The packet time attribute.

Description

Returns the packet time attribute.

10.1.25.3.2 - CSdpFieldAttributePtime::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data. This method has been modified in order to skip some syntax errors when parsing the attribute value. When a parsing error is detected, a default ptime value is used, the rest of the line is skipped, and the function returns no error.

10.1.25.3.3 - CSdpFieldAttributePtime::Reset Method

Resets this object.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 179).

10.1.25.3.4 - CSdpFieldAttributePtime::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.25.3.5 - CSdpFieldAttributePtime::SetPacketTime Method

Sets the packet time.

C++

```
void SetPacketTime(IN int32_t nPacketTime);
```

Parameters

Parameters	Description
`IN int32_t nPacketTime`	The packet time attribute.

Description

Sets the packet time attribute.

10.1.25.3.6 - CSdpFieldAttributePtime::Validate Method

Returns true if data members are valid.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.25.4 - Operators

10.1.25.4.1 - CSdpFieldAttributePtime::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributePtime& operator =(IN const CSdpFieldAttributePtime& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributePtime& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.25.4.2 - CSdpFieldAttributePtime::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributePtime& rFrom) const;
```

Returns

true if attributes are identical.

Description

Comparison operator

10.1.26 - CSdpFieldAttributeRtcp Class

This class implements an abstraction of the rtcp SDP attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeRtcp

C++

```
class CSdpFieldAttributeRtcp : public CSdpParser;
```

Description

This class is an abstraction of the rtcp SDP attribute, as described in RFC 3605. This complies to the following ABNF (using rules in RFC 2327, Appendix A):

```
rtcp-attribute =      "a=rtcp:" port  [nettype space addrtype space
                      connection-address] CRLF

port =                1*(DIGIT)
                      ;should in the range "1024" to "65535" inclusive
                      ;for UDP based media

nettype =             "IN"
                      ;list to be extended

addrtype =            "IP4" | "IP6"
                      ;list to be extended

connection-address =  multicast-address
                      | addr

multicast-address =   3*(decimal-uchar ".") decimal-uchar "/" ttl
                      [ "/" integer ]
                      ;multicast addresses may be in the range
                      ;224.0.0.0 to 239.255.255.255

addr =                FQDN | unicast-address
```

```
unicast-address =    IP4-address | IP6-address

IP4-address =        b1 "." decimal-uchar "." decimal-uchar "." b4

b1 =                 decimal-uchar
                   ;less than "224"; not "0" or "127"
b4 =                 decimal-uchar
                   ;not "0"

IP6-address =        ;to be defined

decimal-uchar =      DIGIT
                   | POS-DIGIT DIGIT
                   | ("1" 2*(DIGIT))
                   | ("2" ("0"|"1"|"2"|"3"|"4") DIGIT)
                   | ("2" "5" ("0"|"1"|"2"|"3"|"4"|"5"))

integer =            POS-DIGIT *(DIGIT)

alpha-numeric =      ALPHA | DIGIT

space =              %d32

ttl =                decimal-uchar
```

Location

SdpParser/CSdpFieldAttributeRtcp.h

See Also

CSdpParser (see page 351)

Constructors

Constructor	Description
◆ CSdpFieldAttributeRtcp (see page 183)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeRtcp (see page 183)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 187)	Assignment operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetAddress (see page 184)	Gets the RTCP address.
◆ GetAddressTypeId (see page 184)	Gets the address type ID.
◆ GetAddressTypeString (see page 184)	Gets the address type string.
◆ GetNetworkTypeId (see page 184)	Gets the network type ID.
◆ GetNetworkTypeString (see page 184)	Gets the network type string.
◆ GetPort (see page 185)	Gets the RTCP port.
◆ Parse (see page 185)	Parses rpszStartPosition wrt rtcp media attribute syntax.
◆ V Reset (see page 185)	Resets the data in the parser.
◆ Serialize (see page 185)	Serializes the data member into the proper rtcp media attribute syntax.
◆ SetAddress (see page 186)	Sets the RTCP address.
◆ SetAddressTypeId (see page 186)	Sets the address type ID.
◆ SetAddressTypeString (see page 186)	Sets the address type string.
◆ SetNetworkTypeId (see page 186)	Sets the network type ID.
◆ SetNetworkTypeString (see page 186)	Sets the network type string.
◆ SetPort (see page 187)	Sets the RTCP port.
◆ V Validate (see page 187)	Validates the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.26.1 - Constructors

10.1.26.1.1 - CSdpFieldAttributeRtcp::CSdpFieldAttributeRtcp Constructor

Default constructor.

C++

```
CSdpFieldAttributeRtcp();
```

Description

Default constructor.

10.1.26.2 - Destructors

10.1.26.2.1 - CSdpFieldAttributeRtcp::~CSdpFieldAttributeRtcp Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeRtcp();
```

Description

Destructor.

10.1.26.3 - Methods

10.1.26.3.1 - CSdpFieldAttributeRtcp::GetAddress Method

Gets the RTCP address.

C++

```
const char* GetAddress() const;
```

Returns

The rtcp attribute address.

Description

Returns the rtcp attribute address.

10.1.26.3.2 - CSdpFieldAttributeRtcp::GetAddressTypeId Method

Gets the address type ID.

C++

```
EAddressType GetAddressTypeId() const;
```

Returns

The rtcp attribute CSdpParser::EAddressType.

Description

Returns the address type ID.

10.1.26.3.3 - CSdpFieldAttributeRtcp::GetAddressTypeString Method

Gets the address type string.

C++

```
const char* GetAddressTypeString() const;
```

Returns

The address type string.

Description

Returns the address type string.

10.1.26.3.4 - CSdpFieldAttributeRtcp::GetNetworkTypeId Method

Gets the network type ID.

C++

```
ENetworkType GetNetworkTypeId() const;
```

Returns

The rtcp attribute CSdpParser::ENetworkType.

Description

Returns the network type ID.

10.1.26.3.5 - CSdpFieldAttributeRtcp::GetNetworkTypeString Method

Gets the network type string.

C++

```
const char* GetNetworkTypeString() const;
```

Returns

The network type string.

Description

Returns the network type string.

10.1.26.3.6 - CSdpFieldAttributeRtcp::GetPort Method

Gets the RTCP port.

C++

```
int32_t GetPort() const;
```

Returns

The rtcp attribute port.

Description

Returns the rtcp attribute port.

10.1.26.3.7 - CSdpFieldAttributeRtcp::Parse Method

Parses rpszStartPosition wrt rtcp media attribute syntax.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	resS_OK: parsing is successful.
`resFE_INVALID_ARGUMENT`	there is a syntax error in the attribute

Returns

eERROR: syntax error found during parsing results from CSdpParser::GetToken()

Description

Parses rpszStartPosition wrt rtcp media attribute syntax. A port value of -1 means the attribute is absent from the media announcement.

See Also

CSdpParser::GetToken()

10.1.26.3.8 - CSdpFieldAttributeRtcp::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Sets the string representations to an empty string, the port to zero and the network and address type to unknown.

10.1.26.3.9 - CSdpFieldAttributeRtcp::Serialize Method

Serializes the data member into the proper rtcp media attribute syntax.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The serializing result container.

Description

Serializes the data member into the proper rtcp media attribute syntax.

10.1.26.3.10 - CSdpFieldAttributeRtcp::SetAddress Method

Sets the RTCP address.

C++

```
void SetAddress(IN const char* pszAddress);
```

Parameters

Parameters	Description
`IN const char* pszAddress`	The address to set.

Description

Sets the address.

10.1.26.3.11 - CSdpFieldAttributeRtcp::SetAddressTypeId Method

Sets the address type ID.

C++

```
void SetAddressTypeId(IN EAddressType eAddressType);
```

Parameters

Parameters	Description
`IN EAddressType eAddressType`	The EAddressType to set.

Description

Sets the address type ID.

10.1.26.3.12 - CSdpFieldAttributeRtcp::SetAddressTypeString Method

Sets the address type string.

C++

```
void SetAddressTypeString(IN const char* pszAddressType);
```

Parameters

Parameters	Description
`IN const char* pszAddressType`	The address type string to set.

Description

Sets the address type string.

10.1.26.3.13 - CSdpFieldAttributeRtcp::SetNetworkTypeId Method

Sets the network type ID.

C++

```
void SetNetworkTypeId(IN ENetworkType eNetworkType);
```

Parameters

Parameters	Description
`IN ENetworkType eNetworkType`	The ENetworkType to set.

Description

Sets the network type ID.

10.1.26.3.14 - CSdpFieldAttributeRtcp::SetNetworkTypeString Method

Sets the network type string.

C++

```
void SetNetworkTypeString(IN const char* pszNetworkType);
```

Parameters

Parameters	Description
`IN const char* pszNetworkType`	The network type string to set.

Description

Sets the network type string.

10.1.26.3.15 - CSdpFieldAttributeRtcp::SetPort Method

Sets the RTCP port.

C++

```
void SetPort(IN uint16_t uPort);
```

Parameters

Parameters	Description
`IN uint16_t uPort`	The port to set.

Description

Sets the port of the rtcp sttribute.

10.1.26.3.16 - CSdpFieldAttributeRtcp::Validate Method

Validates the parsed data.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Verifies that the port value is non-zero, and if there is an address specified, validates it. A port value of -1 means the attribute is absent from the media announcement.

See Also

CSocketAddr::IsValid (see page 353)()

10.1.26.4 - Operators

10.1.26.4.1 - CSdpFieldAttributeRtcp::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeRtcp& operator =(IN const CSdpFieldAttributeRtcp& rFrom);
```

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.27 - CSdpFieldAttributeRtpmap Class

This class implements an abstraction of an attribute-rtpmap.

Class Hierarchy

CSdpParser
CSdpFieldAttributeRtpmap

C++

```
class CSdpFieldAttributeRtpmap : public CSdpParser;
```

Description

This class is an abstraction of an attribute-rtpmap in a SDP packet.

The parsing of this attribute-rtpmap does not follow the general BMF described for an attribute in RFC 2327. It follows another BNF described in RFC 2327.

RFC 2327 BNF:

```
attribute-rtpmap    =     "rtpmap:" payload-type encoding-name / clock-rate
                          [/ encoding-parameters]
payload-type        =     space byte-string
encoding-name       =     space byte-string
clock-rate          =     space byte-string
encoding-parameters =     space *(byte-string)
byte-string         =     1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Note that the only encoding-parameter accepted right now is an integer.

Location

SdpParser/CSdpFieldAttributeRtpmap.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeRtpmap (see page 189)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeRtpmap (see page 190)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 193)	Assignment operator.
◆ == (see page 193)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
GetClockRate (see page 190)	Gets the clock rate.
GetEncoding (see page 190)	Gets the compression algorithm.
GetEncodingName (see page 190)	Gets the encoding name.
GetEncodingParameters (see page 190)	Gets the encoding parameters.
GetPayloadType (see page 191)	Gets the payload type.
Parse (see page 191)	Parses all the needed information for this field.
Reset (see page 191)	Resets all the data members.
Serialize (see page 191)	Generates the data blob from the data members.
SetClockRate (see page 192)	Sets the clock rate.
SetEncoding (see page 192)	Sets the compression algorithm.
SetEncodingName (see page 192)	Sets the encoding name.
SetEncodingParameters (see page 192)	Sets the encoding parameters.
SetPayloadType (see page 193)	Sets the payload type.
Validate (see page 193)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
V Reset (see page 353)	Resets the data in the parser.
A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.27.1 - Constructors

10.1.27.1.1 - CSdpFieldAttributeRtpmap

10.1.27.1.1.1 - CSdpFieldAttributeRtpmap::CSdpFieldAttributeRtpmap Constructor

Default constructor.

C++

```
CSdpFieldAttributeRtpmap();
```

Description

Constructor

10.1.27.1.1.2 - CSdpFieldAttributeRtpmap::CSdpFieldAttributeRtpmap Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeRtpmap(IN const CSdpFieldAttributeRtpmap& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeRtpmap& rFrom`	The object to be copied.

Description

Copy constructor

10.1.27.2 - Destructors

10.1.27.2.1 - CSdpFieldAttributeRtpmap::~CSdpFieldAttributeRtpmap Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeRtpmap();
```

Description

Destructor

10.1.27.3 - Methods

10.1.27.3.1 - CSdpFieldAttributeRtpmap::GetClockRate Method

Gets the clock rate.

C++

```
int32_t GetClockRate() const;
```

Returns

The clock rate.

Description

Returns the rtp map attribute clock rate.

10.1.27.3.2 - CSdpFieldAttributeRtpmap::GetEncoding Method

Gets the compression algorithm.

C++

```
CSdpParser::ERtpCompressionAlgorithm GetEncoding() const;
```

Returns

The compression algorithm used.

Description

Returns the rtp map attribute ERtpCompressionAlgorithm used.

10.1.27.3.3 - CSdpFieldAttributeRtpmap::GetEncodingName Method

Gets the encoding name.

C++

```
const char* GetEncodingName() const;
```

Returns

The encoding name.

Description

Returns the rtp map attribute encoding name.

10.1.27.3.4 - CSdpFieldAttributeRtpmap::GetEncodingParameters Method

Gets the encoding parameters.

C++

```
int32_t GetEncodingParameters() const;
```

Returns

The encoding parameters

Description

Returns the rtp map attribute encoding parameters.

10.1.27.3.5 - CSdpFieldAttributeRtpmap::GetPayloadType Method

Gets the payload type.

C++

```
int32_t GetPayloadType() const;
```

Returns

The payload type

Description

Returns the rtp map attribute payload type.

10.1.27.3.6 - CSdpFieldAttributeRtpmap::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.27.3.7 - CSdpFieldAttributeRtpmap::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 191).

10.1.27.3.8 - CSdpFieldAttributeRtpmap::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.27.3.9 - CSdpFieldAttributeRtpmap::SetClockRate Method

Sets the clock rate.

C++

```
void SetClockRate(IN int32_t nClockRate);
```

Parameters

Parameters	Description
`IN int32_t nClockRate`	The clock rate to set.

Description

Sets the rtp map attribute clock rate.

10.1.27.3.10 - CSdpFieldAttributeRtpmap::SetEncoding Method

Sets the compression algorithm.

C++

```
void SetEncoding(IN CSdpParser::ERtpCompressionAlgorithm eEncoding);
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpCompressionAlgorithm eEncoding`	The compression algorithm to set.

Description

Sets the rtp map attribute compression algorithm and encoding name.

See Also

SetEncodingName (see page 192)

10.1.27.3.11 - CSdpFieldAttributeRtpmap::SetEncodingName Method

Sets the encoding name.

C++

```
void SetEncodingName(IN const char* pszEncodingName);
```

Parameters

Parameters	Description
`IN const char* pszEncodingName`	The encoding name to set.

Description

Sets the rtp map attribute encoding name and compression algorithm.

See Also

SetEncoding (see page 192)

10.1.27.3.12 - CSdpFieldAttributeRtpmap::SetEncodingParameters Method

Sets the encoding parameters.

C++

```
void SetEncodingParameters(IN int32_t nEncodingParameters);
```

Parameters

Parameters	Description
`IN int32_t nEncodingParameters`	The encoding parameters to set.

Description

Sets the rtp map attribute encoding parameters.

10.1.27.3.13 - CSdpFieldAttributeRtpmap::SetPayloadType Method

Sets the payload type.

C++

```
void SetPayloadType(IN int32_t nPayloadType);
```

Parameters

Parameters	Description
`IN int32_t nPayloadType`	The payload type to set.

Description

Sets the rtp map attribute payload type.

10.1.27.3.14 - CSdpFieldAttributeRtpmap::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.27.4 - Operators

10.1.27.4.1 - CSdpFieldAttributeRtpmap::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeRtpmap& operator =(IN const CSdpFieldAttributeRtpmap& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeRtpmap& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.27.4.2 - CSdpFieldAttributeRtpmap::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeRtpmap& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeRtpmap& rFrom`	The CSdpFieldAttributeRtpmap (see page 187) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.28 - CSdpFieldAttributeSilenceSupp Class

This class implements the parsing and serializing of the silence suppression attribute.

Class Hierarchy

CSdpParser
CSdpFieldAttributeSilenceSupp

C++

```
class CSdpFieldAttributeSilenceSupp : public CSdpParser;
```

Description

This class is used to handle the parsing and serializing of the silence suppression attribute defined in RFC 3108.

The attribute follows the following ABNF: a=silenceSupp: <silenceSuppEnable> <silenceTimer> <suppPref> <sidUse> <fxnslevel>

The validation of this attribute is done only on the on/off part of the attriute. The rest of the validation is done by the application itself.

Location

SdpParser/CSdpFieldAttributeSilenceSupp.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeSilenceSupp (see page 195)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeSilenceSupp (see page 195)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 197)	Assignment Operator.
◆ == (see page 198)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetValue (see page 196)	Gets the attribute string.

◆ IsOn (see page 196)	Verifies if the attribute is on.
◆ Parse (see page 196)	Parses the data. Can return any type of EParserResult.
◆ V Reset (see page 196)	Resets the data in the parser.
◆ Serialize (see page 196)	Serializes the data to a CBlob.
◆ SetValue (see page 197)	Sets the attribute on or off.
◆ V Validate (see page 197)	Validates and checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

10.1.28.1 - Constructors

10.1.28.1.1 - CSdpFieldAttributeSilenceSupp

10.1.28.1.1.1 - CSdpFieldAttributeSilenceSupp::CSdpFieldAttributeSilenceSupp Constructor

Default constructor.

C++

```
CSdpFieldAttributeSilenceSupp();
```

Description

Default constructor.

10.1.28.1.1.2 - CSdpFieldAttributeSilenceSupp::CSdpFieldAttributeSilenceSupp Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeSilenceSupp(IN const CSdpFieldAttributeSilenceSupp& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeSilenceSupp& rSrc`	The CSdpFieldAttributeSilenceSupp from which to construct.

Description

Copy constructor.

10.1.28.2 - Destructors

10.1.28.2.1 - CSdpFieldAttributeSilenceSupp::~CSdpFieldAttributeSilenceSupp Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeSilenceSupp();
```

Description

Destructor.

10.1.28.3 - Methods

10.1.28.3.1 - CSdpFieldAttributeSilenceSupp::GetValue Method

Gets the attribute string.

C++

```
const char* GetValue() const;
```

Returns

szValue: The string set to the attribute value.

Description

Gets the string set in the attribute.

10.1.28.3.2 - CSdpFieldAttributeSilenceSupp::IsOn Method

Verifies if the attribute is on.

C++

```
bool IsOn() const;
```

Returns

True if the first two characters of the string are equal to "on", false otherwise.

Description

Checks if the silence suppression attribute is supported.

10.1.28.3.3 - CSdpFieldAttributeSilenceSupp::Parse Method

Parses the data. Can return any type of EParserResult.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	The string containing the silence suppression attribute.
`OUT mxt_result& rres`	resS_OK: parsing successful
`resFE_INVALID_ARGUMENT`	string failed to parse.

Returns

Value used to control the parsing. eERROR: an error has occurred eOK_EOL: the line was parsed and an EOL was removed eOK_NULL: the line was parsed and 'rpszPosition' now points to a NULL

Description

Parses the next SDP attribute. Even if parsing is successful, it does not mean that the attribute parsed is valid.

10.1.28.3.4 - CSdpFieldAttributeSilenceSupp::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets the data members.

10.1.28.3.5 - CSdpFieldAttributeSilenceSupp::Serialize Method

Serializes the data to a CBlob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The blob to which append the silence suppression attribute.

Description

Serializes the attribute to the CBlob. It is serialized in the form: "a=silenceSupp:value"

10.1.28.3.6 - SetValue

10.1.28.3.6.1 - CSdpFieldAttributeSilenceSupp::SetValue Method

Sets the attribute on or off.

C++

```
void SetValue(IN bool bOn);
```

Parameters

Parameters	Description
`IN bool bOn`	True to set the attribute to "on", false to set it to "off"

Description

Sets the string to the attribute. If true, the string is set to "on - - - -" and is set to "off - - - -" if false.

10.1.28.3.6.2 - CSdpFieldAttributeSilenceSupp::SetValue Method

Sets the attribute to the specified string.

C++

```
void SetValue(IN const char* szValue);
```

Parameters

Parameters	Description
`IN const char* szValue`	The string to set to the attribute value.

Description

Sets the string to the attribute. If the string does not begin with either "on" or "off", the attribute is invalid. The rest of the string is considered free text.

10.1.28.3.7 - CSdpFieldAttributeSilenceSupp::Validate Method

Validates and checks the validity of the parsed data.

C++

```
virtual bool Validate();
```

Returns

True if the string represents a valide silence supp attribute, false otherwise.

Description

Validates that the string contained in this attribute starts with the string "on" or "off".

10.1.28.4 - Operators

10.1.28.4.1 - CSdpFieldAttributeSilenceSupp::= Operator

Assignment Operator.

C++

```
CSdpFieldAttributeSilenceSupp& operator =(IN const CSdpFieldAttributeSilenceSupp& rSrc);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeSilenceSupp& rSrc`	The CSdpFieldAttributeSilenceSupp (see page 194) to copy.

Returns

The new object.

Description

Sets the data of rSrc to this object.

10.1.28.4.2 - CSdpFieldAttributeSilenceSupp::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeSilenceSupp& rSrc) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeSilenceSupp& rSrc`	The CSdpFieldAttributeSilenceSupp (see page 194) with which to compare.

Returns

True if both objects are equal, false otherwise.

Description

Comparison operator. Validates that parameter is equal to this object.

10.1.29 - CSdpFieldAttributeT38ErrorControl Class

This class implements an abstraction of an attribute-t38-error-control.

Class Hierarchy

CSdpParser
CSdpFieldAttributeT38ErrorControl

C++

```
class CSdpFieldAttributeT38ErrorControl : public CSdpParser;
```

Description

This class is an abstraction of an attribute-t38-error-control in a SDP packet.

The parsing of this attribute-t38-error-control is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-t38-error-control =   "T38FaxUdpEC:" byte-string
byte-string                 =   1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)

From ITU-T Rec. T.38 (03/2002):

Error Correction
Att-field=T38FaxUdpEC
Att-value = t38UDPFEC | t38UDPRedundancy
```

Location

SdpParser/CSdpFieldAttributeT38ErrorControl.h

Constructors

Constructor	Description
CSdpFieldAttributeT38ErrorControl (see page 200)	Default Constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldAttributeT38ErrorControl (see page 200)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 202)	Assignment operator.
◆ == (see page 202)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetErrorControl (see page 200)	Gets the error control.
◆ Parse (see page 200)	Parses all the needed information for this field.
◆ Reset (see page 201)	Resets all the data members.
◆ Serialize (see page 201)	Generates the data blob from the data members.
◆ SetErrorControl (see page 201)	Sets the error control.
◆ Validate (see page 201)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.29.1 - Constructors

10.1.29.1.1 - CSdpFieldAttributeT38ErrorControl

10.1.29.1.1.1 - CSdpFieldAttributeT38ErrorControl::CSdpFieldAttributeT38ErrorControl Constructor

Default Constructor.

C++

```
CSdpFieldAttributeT38ErrorControl();
```

Description

Constructor

10.1.29.1.1.2 - CSdpFieldAttributeT38ErrorControl::CSdpFieldAttributeT38ErrorControl Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeT38ErrorControl(IN const CSdpFieldAttributeT38ErrorControl& rFrom);
```

10.1.29.2 - Destructors

10.1.29.2.1 - CSdpFieldAttributeT38ErrorControl::~CSdpFieldAttributeT38ErrorControl Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeT38ErrorControl();
```

Description

Destructor

10.1.29.3 - Methods

10.1.29.3.1 - CSdpFieldAttributeT38ErrorControl::GetErrorControl Method

Gets the error control.

C++

```
const char* GetErrorControl() const;
```

Returns

The attribute T38 Error Control.

Description

Returns the T38 Error Control.

10.1.29.3.2 - CSdpFieldAttributeT38ErrorControl::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.29.3.3 - CSdpFieldAttributeT38ErrorControl::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 200).

10.1.29.3.4 - CSdpFieldAttributeT38ErrorControl::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.29.3.5 - CSdpFieldAttributeT38ErrorControl::SetErrorControl Method

Sets the error control.

C++

```
void SetErrorControl(IN const char* pszErrorControl);
```

Parameters

Parameters	Description
`IN const char* pszErrorControl`	The value of the attribute T38ErrorControl to set.

Description

Sets the attribute T38ErrorControl.

10.1.29.3.6 - CSdpFieldAttributeT38ErrorControl::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.29.4 - Operators

10.1.29.4.1 - CSdpFieldAttributeT38ErrorControl::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeT38ErrorControl& operator =(IN const CSdpFieldAttributeT38ErrorControl& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeT38ErrorControl& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.29.4.2 - CSdpFieldAttributeT38ErrorControl::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeT38ErrorControl& rFrom) const;
```

Returns

true if both attributes contain the same t38 error control value.

Description

Comparison operator.

10.1.30 - CSdpFieldAttributeT38FacsimileMaxBuffer Class

This class implements an abstraction of an attribute-t38-facsimile-max-buffer.

Class Hierarchy

CSdpParser
CSdpFieldAttributeT38FacsimileMaxBuffer

C++

```
class CSdpFieldAttributeT38FacsimileMaxBuffer : public CSdpParser;
```

Description

This class is an abstraction of an attribute-t38-facsimile-max-buffer in a SDP packet.

The parsing of this attribute-t38-facsimile-max-buffer is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-t38-facsimile-max-buffer =    "T38FaxMaxBuffer:" byte-string
byte-string                        =    1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeT38FacsimileMaxBuffer.h

Constructors

Constructor	Description
CSdpFieldAttributeT38FacsimileMaxBuffer (see page 203)	Default Constructor.

CSdpParser Class

CSdpParser Class	Description
CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeT38FacsimileMaxBuffer (see page 204)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 205)	Assignment operator.
◆ == (see page 206)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetMaxBuffer (see page 204)	Gets the maximum buffer.
◆ Parse (see page 204)	Parses all the needed information for this field.
◆ Reset (see page 204)	Resets all the data members.
◆ Serialize (see page 205)	Generates the data blob from the data members.
◆ SetMaxBuffer (see page 205)	Sets the maximum buffer.
◆ Validate (see page 205)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.30.1 - Constructors

10.1.30.1.1 - CSdpFieldAttributeT38FacsimileMaxBuffer

10.1.30.1.1.1 - CSdpFieldAttributeT38FacsimileMaxBuffer::CSdpFieldAttributeT38FacsimileMaxBuffer Constructor

Default Constructor.

C++

```
CSdpFieldAttributeT38FacsimileMaxBuffer();
```

Description

Constructor

10.1.30.1.1.2 - CSdpFieldAttributeT38FacsimileMaxBuffer::CSdpFieldAttributeT38FacsimileMaxBuffer Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeT38FacsimileMaxBuffer(IN const CSdpFieldAttributeT38FacsimileMaxBuffer& rFrom);
```

10.1.30.2 - Destructors

10.1.30.2.1 - CSdpFieldAttributeT38FacsimileMaxBuffer::~CSdpFieldAttributeT38FacsimileMaxBuffer Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeT38FacsimileMaxBuffer();
```

Description

Destructor

10.1.30.3 - Methods

10.1.30.3.1 - CSdpFieldAttributeT38FacsimileMaxBuffer::GetMaxBuffer Method

Gets the maximum buffer.

C++

```
const int32_t GetMaxBuffer() const;
```

Returns

The maximum buffer attribute.

Description

Returns the T38 Facsimile maximum buffer.

10.1.30.3.2 - CSdpFieldAttributeT38FacsimileMaxBuffer::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.30.3.3 - CSdpFieldAttributeT38FacsimileMaxBuffer::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 204).

10.1.30.3.4 - CSdpFieldAttributeT38FacsimileMaxBuffer::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.30.3.5 - CSdpFieldAttributeT38FacsimileMaxBuffer::SetMaxBuffer Method

Sets the maximum buffer.

C++

```
void SetMaxBuffer(IN const int nMaxBuffer);
```

Parameters

Parameters	Description
`IN const int nMaxBuffer`	The value of the attribute T38FacsimilieMaxBuffer to set.

Description

Sets the max buffer T38FacsimilieMaxBuffer.

10.1.30.3.6 - CSdpFieldAttributeT38FacsimileMaxBuffer::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.30.4 - Operators

10.1.30.4.1 - CSdpFieldAttributeT38FacsimileMaxBuffer::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeT38FacsimileMaxBuffer& operator =(IN const CSdpFieldAttributeT38FacsimileMaxBuffer& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeT38FacsimileMaxBuffer& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.30.4.2 - CSdpFieldAttributeT38FacsimileMaxBuffer::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeT38FacsimileMaxBuffer& rFrom) const;
```

Returns

true if both attributes contain the same t38 maximum buffer value.

Description

Comparison operator

10.1.31 - CSdpFieldAttributeT38FacsimileRateMgmnt Class

This class implements an abstraction of an attribute-t38-Facsimile-rate-mgmnt.

Class Hierarchy

C++

```
class CSdpFieldAttributeT38FacsimileRateMgmnt : public CSdpParser;
```

Description

This class is an abstraction of an attribute-t38-Facsimile-rate-mgmnt in a SDP packet.

The parsing of this attribute-t38-Facsimile-rate-mgmnt is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-t38-Facsimile-rate-mgmnt = "T38FaxRateManagement:" byte-string
byte-string                        = 1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeT38FacsimileRateMgmnt.h

Constructors

Constructor	Description
CSdpFieldAttributeT38FacsimileRateMgmnt (see page 207)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
~CSdpFieldAttributeT38FacsimileRateMgmnt (see page 208)	Destructor.

CSdpParser Class

CSdpParser Class	Description
~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 209)	Assignment operator.
◆ == (see page 210)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetFacsimileRateMgmnt (see page 208)	Gets facsimile rate management.
◆ Parse (see page 208)	Parses all the needed information for this field.
◆ Reset (see page 208)	Resets all the data members.
◆ Serialize (see page 209)	Generates the data blob from the data members.
◆ SetFacsimileRateMgmnt (see page 209)	Sets facsimile rate management.
◆ Validate (see page 209)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.31.1 - Constructors

10.1.31.1.1 - CSdpFieldAttributeT38FacsimileRateMgmnt

10.1.31.1.1.1 - CSdpFieldAttributeT38FacsimileRateMgmnt::CSdpFieldAttributeT38FacsimileRateMgmnt Constructor

Default constructor.

C++

```
CSdpFieldAttributeT38FacsimileRateMgmnt();
```

Description

Constructor

10.1.31.1.1.2 - CSdpFieldAttributeT38FacsimileRateMgmnt::CSdpFieldAttributeT38FacsimileRateMgmnt Constructor

Copy constructor.

C++

```
CSdpFieldAttributeT38FacsimileRateMgmnt(IN const CSdpFieldAttributeT38FacsimileRateMgmnt& rFrom);
```

10.1.31.2 - Destructors

10.1.31.2.1 - CSdpFieldAttributeT38FacsimileRateMgmnt::~CSdpFieldAttributeT38FacsimileRateMgmnt Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeT38FacsimileRateMgmnt();
```

Description

Destructor

10.1.31.3 - Methods

10.1.31.3.1 - CSdpFieldAttributeT38FacsimileRateMgmnt::GetFacsimileRateMgmnt Method

Gets facsimile rate management.

C++

```
const char* GetFacsimileRateMgmnt() const;
```

Returns

The attribute T38FacsimileRateMgmnt.

Description

Returns the T38FacsimileRateMgmnt attribute.

10.1.31.3.2 - CSdpFieldAttributeT38FacsimileRateMgmnt::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.31.3.3 - CSdpFieldAttributeT38FacsimileRateMgmnt::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 208).

10.1.31.3.4 - CSdpFieldAttributeT38FacsimileRateMgmnt::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.31.3.5 - CSdpFieldAttributeT38FacsimileRateMgmnt::SetFacsimileRateMgmnt Method

Sets facsimile rate management.

C++

```
void SetFacsimileRateMgmnt(IN const char* pszFaxRate);
```

Parameters

Parameters	Description
`IN const char* pszFaxRate`	The value of the attribute T38FacsimileRateMgmnt to set.

Description

Sets the T38FacsimileRateMgmnt attribute.

10.1.31.3.6 - CSdpFieldAttributeT38FacsimileRateMgmnt::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.31.4 - Operators

10.1.31.4.1 - CSdpFieldAttributeT38FacsimileRateMgmnt::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeT38FacsimileRateMgmnt& operator =(IN const CSdpFieldAttributeT38FacsimileRateMgmnt& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeT38FacsimileRateMgmnt& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.31.4.2 - CSdpFieldAttributeT38FacsimileRateMgmnt::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeT38FacsimileRateMgmnt& rFrom) const;
```

Returns

true if both attributes contain the same T38FacsimileRateMgmnt value.

Description

Comparison operator

10.1.32 - CSdpFieldAttributeTranscoding Class

This class implements an abstraction of an attribute-transcoding.

Class Hierarchy

CSdpParser
CSdpFieldAttributeTranscoding

C++

```
class CSdpFieldAttributeTranscoding : public CSdpParser;
```

Description

This class is an abstraction of an attribute-transcoding in a SDP packet.

This class is no longer part of T.38 and is not used by others in the whole SdpParser package. It will eventually be removed.

Location

SdpParser/CSdpFieldAttributeTranscoding.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeTranscoding (see page 211)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeTranscoding (see page 211)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 213)	Assignment operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetTranscoding (see page 212)	Gets the transcoding attribute.
◆ Parse (see page 212)	Parses all the needed information for this field.
◆ Reset (see page 212)	Resets all the data members.
◆ Serialize (see page 212)	Generates the data blob from the data members.
◆ SetTranscoding (see page 213)	Sets the transcoding attribute.
◆ Validate (see page 213)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.32.1 - Constructors

10.1.32.1.1 - CSdpFieldAttributeTranscoding

10.1.32.1.1.1 - CSdpFieldAttributeTranscoding::CSdpFieldAttributeTranscoding Constructor

Default constructor.

C++

```
CSdpFieldAttributeTranscoding();
```

Description

Constructor

10.1.32.1.1.2 - CSdpFieldAttributeTranscoding::CSdpFieldAttributeTranscoding Constructor

Copy Constructor.

C++

```
CSdpFieldAttributeTranscoding(IN const CSdpFieldAttributeTranscoding& rFrom);
```

10.1.32.2 - Destructors

10.1.32.2.1 - CSdpFieldAttributeTranscoding::~CSdpFieldAttributeTranscoding Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeTranscoding();
```

Description

Destructor

10.1.32.3 - Methods

10.1.32.3.1 - CSdpFieldAttributeTranscoding::GetTranscoding Method

Gets the transcoding attribute.

C++

```
const char* GetTranscoding() const;
```

Returns

The transcoding attribute.

Description

Returns the transcoding attribute.

10.1.32.3.2 - CSdpFieldAttributeTranscoding::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.32.3.3 - CSdpFieldAttributeTranscoding::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 212).

10.1.32.3.4 - CSdpFieldAttributeTranscoding::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.32.3.5 - CSdpFieldAttributeTranscoding::SetTranscoding Method

Sets the transcoding attribute.

C++

```
void SetTranscoding(IN const char* pszTranscoding);
```

Parameters

Parameters	Description
`IN const char* pszTranscoding`	The value of the attribute transcoding to set.

Description

Sets the transcoding attribute.

10.1.32.3.6 - CSdpFieldAttributeTranscoding::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.32.4 - Operators

10.1.32.4.1 - CSdpFieldAttributeTranscoding::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeTranscoding& operator =(IN const CSdpFieldAttributeTranscoding& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeTranscoding& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.33 - CSdpFieldAttributeTranscodingJBIG Class

This class implements an abstraction of an attribute-transcoding-jbig.

Class Hierarchy

CSdpParser
CSdpFieldAttributeTranscodingJBIG

C++

```
class CSdpFieldAttributeTranscodingJBIG : public CSdpParser;
```

Description

This class is an abstraction of an attribute-transcoding-jbig field in a SDP packet.

The parsing of this attribute-transcoding-jbig is a specific case of an attribute. The basic BNF that an attribute can have is described in

CSdpFieldAttributeOther (see page 160).

```
attribute-transcoding-jbig =    "T38FaxTranscodingJBIG:" [bit]
bit                        =    space "0" / "1"
```

Location

SdpParser/CSdpFieldAttributeTranscodingJBIG.h

Constructors

Constructor	Description
CSdpFieldAttributeTranscodingJBIG (see page 215)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
~CSdpFieldAttributeTranscodingJBIG (see page 215)	Destructor.

CSdpParser Class

CSdpParser Class	Description
~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
= (see page 218)	Assignment operator.
== (see page 218)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
= (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
IsImplicitTranscodingJBIG (see page 215)	Indicates if the T38FaxTranscodingJBIG attribute is encoded with the implicit method or the explicit method.
IsTranscodingJBIG (see page 216)	Indicates if the T38FaxTranscodingJBIG attribute is enabled or disabled.
Parse (see page 216)	Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.
Reset (see page 216)	Resets all the data members, to be ready for another call to Parse (see page 216). Disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit.
Serialize (see page 216)	Generates the data blob from the data members.
SetExplicitTranscodingJBIG (see page 217)	Enables or disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 217)(bool) and SetTranscodingJBIG (see page 217)(bool) methods.
SetImplicitEncoding (see page 217)	Sets the encoding method for the T38FaxTranscodingJBIG attribute.
SetTranscodingJBIG (see page 217)	Enables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetTranscodingJBIG(bool) method.
Validate (see page 218)	Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (↗see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (↗see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (↗see page 353)	Resets the data in the parser.
◆ A Validate (↗see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.33.1 - Constructors

10.1.33.1.1 - CSdpFieldAttributeTranscodingJBIG

10.1.33.1.1.1 - CSdpFieldAttributeTranscodingJBIG::CSdpFieldAttributeTranscodingJBIG Constructor

Default constructor.

C++

```
CSdpFieldAttributeTranscodingJBIG();
```

Description

Constructor

10.1.33.1.1.2 - CSdpFieldAttributeTranscodingJBIG::CSdpFieldAttributeTranscodingJBIG Constructor

Copy constructor.

C++

```
CSdpFieldAttributeTranscodingJBIG(IN const CSdpFieldAttributeTranscodingJBIG& rFrom);
```

10.1.33.2 - Destructors

10.1.33.2.1 - CSdpFieldAttributeTranscodingJBIG::~CSdpFieldAttributeTranscodingJBIG Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeTranscodingJBIG();
```

Description

Destructor

10.1.33.3 - Methods

10.1.33.3.1 - CSdpFieldAttributeTranscodingJBIG::IsImplicitTranscodingJBIG Method

Indicates if the T38FaxTranscodingJBIG attribute is encoded with the implicit method or the explicit method.

C++

```
bool IsImplicitTranscodingJBIG() const;
```

Returns

True if the T38FaxTranscodingJBIG attribute is encoded with the implicit method. False if the T38FaxTranscodingJBIG attribute is encoded with the explicit method.

Description

Indicates if the T38FaxTranscodingJBIG attribute is encoded with the implicit method or the explicit method.

10.1.33.3.2 - CSdpFieldAttributeTranscodingJBIG::IsTranscodingJBIG Method

Indicates if the T38FaxTranscodingJBIG attribute is enabled or disabled.

C++

```
bool IsTranscodingJBIG() const;
```

Returns

True if the T38FaxTranscodingJBIG attribute is enabled. False otherwise.

Description

Indicates if the T38FaxTranscodingJBIG attribute is enabled or disabled.

10.1.33.3.3 - CSdpFieldAttributeTranscodingJBIG::Parse Method

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

Compatibility: For backward compatibility, the value string can be present or not.

Example

a=T38FaxTranscodingJBIG <= Is supported and considered to be implicitly true a=T38FaxTranscodingJBIG:0 <= Is supported and considered to be false a=T38FaxTranscodingJBIG:1 <= Is supported and considered to be true

10.1.33.3.4 - CSdpFieldAttributeTranscodingJBIG::Reset Method

Resets all the data members, to be ready for another call to Parse (see page 216). Disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 216). Disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit.

10.1.33.3.5 - CSdpFieldAttributeTranscodingJBIG::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.33.3.6 - CSdpFieldAttributeTranscodingJBIG::SetExplicitTranscodingJBIG Method

Enables or disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 217)(bool) and SetTranscodingJBIG (see page 217)(bool) methods.

C++

```
void SetExplicitTranscodingJBIG(IN bool bSupported);
```

Parameters

Parameters	Description
`IN bool bSupported`	Indicates if the T38FaxTranscodingJBIG attribute is enabled or disabled.

Description

Enables or disables the T38FaxTranscodingJBIG attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 217)(bool) and SetTranscodingJBIG (see page 217)(bool) methods.

10.1.33.3.7 - CSdpFieldAttributeTranscodingJBIG::SetImplicitEncoding Method

Sets the encoding method for the T38FaxTranscodingJBIG attribute.

C++

```
void SetImplicitEncoding(bool bImplicitEncoding);
```

Parameters

Parameters	Description
`bool bImplicitEncoding`	Indicates if the T38FaxTranscodingJBIG attribute is encoded with the implicit method (true) or the explicit method (false).

Description

Sets the encoding method for the T38FaxTranscodingJBIG attribute.

10.1.33.3.8 - SetTranscodingJBIG

10.1.33.3.8.1 - CSdpFieldAttributeTranscodingJBIG::SetTranscodingJBIG Method

Enables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetTranscodingJBIG(bool) method.

C++

```
void SetTranscodingJBIG();
```

Description

Enables the T38FaxTranscodingJBIG attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetTranscodingJBIG(bool) method.

10.1.33.3.8.2 - CSdpFieldAttributeTranscodingJBIG::SetTranscodingJBIG Method

Enables or disables the T38FaxTranscodingJBIG attribute.

C++

```
void SetTranscodingJBIG(bool bEnable);
```

Parameters

Parameters	Description
`bool bEnable`	Indicates if the T38FaxTranscodingJBIG attribute is enabled or disabled.

Description

Enables or disables the T38FaxTranscodingJBIG attribute.

10.1.33.3.9 - CSdpFieldAttributeTranscodingJBIG::Validate Method

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.33.4 - Operators

10.1.33.4.1 - CSdpFieldAttributeTranscodingJBIG::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeTranscodingJBIG& operator =(IN const CSdpFieldAttributeTranscodingJBIG& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeTranscodingJBIG& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.33.4.2 - CSdpFieldAttributeTranscodingJBIG::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeTranscodingJBIG& rFrom) const;
```

Returns

true if both attributes contain the same transcoding jbig value.

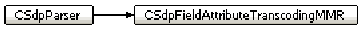
Description

Comparison operator

10.1.34 - CSdpFieldAttributeTranscodingMMR Class

This class implements an abstraction of a attribute-transcoding-mmr.

Class Hierarchy



C++

```
class CSdpFieldAttributeTranscodingMMR : public CSdpParser;
```

Description

This class is an abstraction of a attribute-transcoding-mmr field in a SDP packet.

The parsing of this attribute-transcoding-mmr is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-transcoding-mmr = "T38FaxTranscodingMMR:" [bit]
bit                       =   space "0" / "1"
```

Location

SdpParser/CSdpFieldAttributeTranscodingMMR.h

Constructors

Constructor	Description
CSdpFieldAttributeTranscodingMMR (see page 220)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
~CSdpFieldAttributeTranscodingMMR (see page 220)	Destructor.

CSdpParser Class

CSdpParser Class	Description
~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
= (see page 223)	Assignment operator.
== (see page 223)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
= (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
IsImplicitTranscodingMMR (see page 221)	Indicates if the T38FaxTranscodingMMR attribute is encoded with the implicit method or the explicit method.
IsTranscodingMMR (see page 221)	Indicates if the T38FaxTranscodingMMR attribute is enabled or disabled.
Parse (see page 221)	Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.
Reset (see page 221)	Resets all the data members, to be ready for another call to Parse (see page 221). Disables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit.
Serialize (see page 222)	Generates the data blob from the data members.
SetExplicitTranscodingMMR (see page 222)	Enables or disables the T38FaxTranscodingMMR attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 222)(bool) and SetTranscodingMMR (see page 222)(bool) methods.

SetImplicitEncoding (see page 222)	Sets the encoding method for the T38FaxTranscodingMMR attribute.
SetTranscodingMMR (see page 222)	Enables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetTranscodingMMR(bool) method.
Validate (see page 223)	Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

CSdpParser Class

CSdpParser Class	Description
IsValid (see page 353)	Returns true if the data was parsed successfully.
Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
Reset (see page 353)	Resets the data in the parser.
Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.34.1 - Constructors

10.1.34.1.1 - CSdpFieldAttributeTranscodingMMR

10.1.34.1.1.1 - CSdpFieldAttributeTranscodingMMR::CSdpFieldAttributeTranscodingMMR Constructor

Default constructor.

C++

```
CSdpFieldAttributeTranscodingMMR();
```

Description

Constructor

10.1.34.1.1.2 - CSdpFieldAttributeTranscodingMMR::CSdpFieldAttributeTranscodingMMR Constructor

Copy constructor.

C++

```
CSdpFieldAttributeTranscodingMMR(IN const CSdpFieldAttributeTranscodingMMR& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeTranscodingMMR& rFrom`	The object to be copied.

Description

Copy constructor

10.1.34.2 - Destructors

10.1.34.2.1 - CSdpFieldAttributeTranscodingMMR::~CSdpFieldAttributeTranscodingMMR Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeTranscodingMMR();
```

Description

Destructor

10.1.34.3 - Methods

10.1.34.3.1 - CSdpFieldAttributeTranscodingMMR::IsImplicitTranscodingMMR Method

Indicates if the T38FaxTranscodingMMR attribute is encoded with the implicit method or the explicit method.

C++

```
bool IsImplicitTranscodingMMR() const;
```

Returns

True if the T38FaxTranscodingMMR attribute is encoded with the implicit method. False if the T38FaxTranscodingMMR attribute is encoded with the explicit method.

Description

Indicates if the T38FaxTranscodingMMR attribute is encoded with the implicit method or the explicit method.

10.1.34.3.2 - CSdpFieldAttributeTranscodingMMR::IsTranscodingMMR Method

Indicates if the T38FaxTranscodingMMR attribute is enabled or disabled.

C++

```
bool IsTranscodingMMR() const;
```

Returns

True if the T38FaxTranscodingMMR attribute is enabled. False otherwise.

Description

Indicates if the T38FaxTranscodingMMR attribute is enabled or disabled.

10.1.34.3.3 - CSdpFieldAttributeTranscodingMMR::Parse Method

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

Compatibility: For backward compatibility, the value string can be present or not.

Example

a=T38FaxTranscodingMMR <= Is supported and considered to be implicitly true a=T38FaxTranscodingMMR:0 <= Is supported and considered to be false a=T38FaxTranscodingMMR:1 <= Is supported and considered to be true

10.1.34.3.4 - CSdpFieldAttributeTranscodingMMR::Reset Method

Resets all the data members, to be ready for another call to Parse (see page 221). Disables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 221). Disables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit.

10.1.34.3.5 - CSdpFieldAttributeTranscodingMMR::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.34.3.6 - CSdpFieldAttributeTranscodingMMR::SetExplicitTranscodingMMR Method

Enables or disables the T38FaxTranscodingMMR attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 222)(bool) and SetTranscodingMMR (see page 222)(bool) methods.

C++

```
void SetExplicitTranscodingMMR(IN bool bSupported);
```

Parameters

Parameters	Description
`IN bool bSupported`	Indicates if the T38FaxTranscodingMMR attribute is enabled or disabled.

Description

Enables or disables the T38FaxTranscodingMMR attribute. Sets the encoding method to explicit. This method was deprecated. Use the SetImplicitEncoding (see page 222)(bool) and SetTranscodingMMR (see page 222)(bool) methods.

10.1.34.3.7 - CSdpFieldAttributeTranscodingMMR::SetImplicitEncoding Method

Sets the encoding method for the T38FaxTranscodingMMR attribute.

C++

```
void SetImplicitEncoding(bool bImplicitEncoding);
```

Parameters

Parameters	Description
`bool bImplicitEncoding`	Indicates if the T38FaxTranscodingMMR attribute is encoded with the implicit method (true) or the explicit method (false).

Description

Sets the encoding method for the T38FaxTranscodingMMR attribute.

10.1.34.3.8 - SetTranscodingMMR

10.1.34.3.8.1 - CSdpFieldAttributeTranscodingMMR::SetTranscodingMMR Method

Enables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit. This method was deprecated. Use the SetTranscodingMMR(bool) method.

C++

```
void SetTranscodingMMR();
```

Description

Enables the T38FaxTranscodingMMR attribute. Sets the encoding method to implicit. This method was deprecated. Use the

SetTranscodingMMR(bool) method.

10.1.34.3.8.2 - CSdpFieldAttributeTranscodingMMR::SetTranscodingMMR Method

Enables or disables the T38FaxTranscodingMMR attribute.

C++

```
void SetTranscodingMMR(bool bEnable);
```

Parameters

Parameters	Description
`bool bEnable`	Indicates if the T38FaxTranscodingMMR attribute is enabled or disabled.

Description

Enables or disables the T38FaxTranscodingMMR attribute.

10.1.34.3.9 - CSdpFieldAttributeTranscodingMMR::Validate Method

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.34.4 - Operators

10.1.34.4.1 - CSdpFieldAttributeTranscodingMMR::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeTranscodingMMR& operator =(IN const CSdpFieldAttributeTranscodingMMR& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeTranscodingMMR& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.34.4.2 - CSdpFieldAttributeTranscodingMMR::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeTranscodingMMR& rFrom) const;
```

Returns

true if both attributes contain the same transcoding mmr value.

Description

Comparison operator

10.1.35 - CSdpFieldAttributeVersion Class

This class implements an abstraction of a attribute-version.

Class Hierarchy

CSdpParser
CSdpFieldAttributeVersion

C++

```
class CSdpFieldAttributeVersion : public CSdpParser;
```

Description

This class is an abstraction of a attribute-version in a SDP packet.

The parsing of this attribute-version is a specific case of an attribute. The basic BNF that an attribute can have is described in CSdpFieldAttributeOther (see page 160).

```
attribute-version   = "T38FaxVersion:" byte-string
byte-string         = 1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldAttributeVersion.h

Constructors

Constructor	Description
◆ CSdpFieldAttributeVersion (see page 225)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldAttributeVersion (see page 225)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 227)	Assignment operator.
◆ == (see page 227)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetVersion (see page 226)	Gets the version attribute.
◆ Parse (see page 226)	Parses all the needed information for this field.
◆ Reset (see page 226)	Resets all the data members.

◆ Serialize (see page 226)	Generates the data blob from the data members.
◆ SetVersion (see page 226)	Sets the version attribute.
◆ Validate (see page 227)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.35.1 - Constructors

10.1.35.1.1 - CSdpFieldAttributeVersion

10.1.35.1.1.1 - CSdpFieldAttributeVersion::CSdpFieldAttributeVersion Constructor

Default constructor.

C++

```
CSdpFieldAttributeVersion();
```

Description

Constructor

10.1.35.1.1.2 - CSdpFieldAttributeVersion::CSdpFieldAttributeVersion Constructor

Copy constructor.

C++

```
CSdpFieldAttributeVersion(IN const CSdpFieldAttributeVersion& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeVersion& rFrom`	The object to be copied.

Description

Copy constructor

10.1.35.2 - Destructors

10.1.35.2.1 - CSdpFieldAttributeVersion::~CSdpFieldAttributeVersion Destructor

Destructor.

C++

```
virtual ~CSdpFieldAttributeVersion();
```

Description

Destructor

10.1.35.3 - Methods

10.1.35.3.1 - CSdpFieldAttributeVersion::GetVersion Method

Gets the version attribute.

C++

```
const int32_t GetVersion() const;
```

Returns

The version attribute.

Description

Returns the version attribute.

10.1.35.3.2 - CSdpFieldAttributeVersion::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.35.3.3 - CSdpFieldAttributeVersion::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 226).

10.1.35.3.4 - CSdpFieldAttributeVersion::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.35.3.5 - CSdpFieldAttributeVersion::SetVersion Method

Sets the version attribute.

C++

```
void SetVersion(IN const int32_t nVersion);
```

Parameters

Parameters	Description
`IN const int32_t nVersion`	The version attribute to set.

Description

Sets the version attribute.

10.1.35.3.6 - CSdpFieldAttributeVersion::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.35.4 - Operators

10.1.35.4.1 - CSdpFieldAttributeVersion::= Operator

Assignment operator.

C++

```
CSdpFieldAttributeVersion& operator =(IN const CSdpFieldAttributeVersion& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeVersion& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.35.4.2 - CSdpFieldAttributeVersion::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldAttributeVersion& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeVersion& rFrom`	The CSdpFieldAttributeVersion (see page 224) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.36 - CSdpFieldConnectionData Class

This class implements an abstraction of a connection-field.

Class Hierarchy

CSdpParser
CSdpFieldConnectionData

C++

```
class CSdpFieldConnectionData : public CSdpParser;
```

Description

This class is an abstraction of a connection-field in a SDP packet.

The "c=" field contains connection data. It follows the BNF described in RFC 2327.

RFC 2327 BNF:

```
connection-field    =   ["c=" nettype space addrtype space
                        connection-address CRLF]
nettype             =   "IN"
addrtype            =   "IP4" | "IP6"
connection-address  =   multicast-address
                        | addr
addr                =   FQDN | unicast-address
FQDN                =   4*(alpha-numeric|"-"|".")
unicast-address     =   IP4-address | IP6-address
IP4-address         =   b1 "." decimal-uchar "." decimal-uchar "." b4
b1                  =   decimal-uchar
b4                  =   decimal-uchar
multicast-address   =   3*(decimal-uchar ".") decimal-uchar "/" ttl
                        [ "/" integer ]
ttl                 =   decimal-uchar
space               =   " "
alpha-numeric       =   ALPHA | DIGIT
decimal-uchar       =   DIGIT
                        | POS-DIGIT DIGIT
                        | ("1" 2*(DIGIT))
                        | ("2" ("0"|"1"|"2"|"3"|"4") DIGIT)
                        | ("2" "5" ("0"|"1"|"2"|"3"|"4"|"5"))
integer             =   POS-DIGIT *(DIGIT)
POS-DIGIT           =   "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
```

Location

SdpParser/CSdpFieldConnectionData.h

Constructors

Constructor	Description
◆ CSdpFieldConnectionData (see page 229)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldConnectionData (see page 230)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ != (see page 234)	Comparison operator.
◆ = (see page 234)	Assignment operator.
◆ == (see page 235)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetAddress (see page 230)	Gets the address.
◆ GetAddressTypeId (see page 230)	Gets the address type ID.
◆ GetAddressTypeString (see page 231)	Gets the address type string.
◆ GetNbAddresses (see page 231)	Gets the number of addresses.
◆ GetNetworkTypeId (see page 231)	Gets the network type ID.
◆ GetNetworkTypeString (see page 231)	Gets the network type string.
◆ GetTtl (see page 231)	Gets TTL.
◆ Parse (see page 232)	Parses all the needed information for this field.
◆ Reset (see page 232)	Resets all the data members.
◆ Serialize (see page 232)	Generates the data blob from the data members.
◆ SetAddress (see page 232)	Sets the address.
◆ SetAddressTypeId (see page 233)	Sets the address type ID.
◆ SetAddressTypeString (see page 233)	Sets the address type string.
◆ SetNbAddresses (see page 233)	Sets the number of addresses.
◆ SetNetworkTypeId (see page 233)	Sets the network type ID.
◆ SetNetworkTypeString (see page 233)	Sets the network type string.
◆ SetTtl (see page 234)	Sets TTL.
◆ Validate (see page 234)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.36.1 - Constructors

10.1.36.1.1 - CSdpFieldConnectionData

10.1.36.1.1.1 - CSdpFieldConnectionData::CSdpFieldConnectionData Constructor

Default constructor.

C++

```
CSdpFieldConnectionData();
```

Description

Constructor

10.1.36.1.1.2 - CSdpFieldConnectionData::CSdpFieldConnectionData Constructor

Copy constructor.

C++

```
CSdpFieldConnectionData(IN const CSdpFieldConnectionData& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rFrom`	The object to be copied.

Description

Copy constructor

10.1.36.2 - Destructors

10.1.36.2.1 - CSdpFieldConnectionData::~CSdpFieldConnectionData Destructor

Destructor.

C++

```
virtual ~CSdpFieldConnectionData();
```

Description

Destructor

10.1.36.3 - Methods

10.1.36.3.1 - CSdpFieldConnectionData::GetAddress Method

Gets the address.

C++

```
const char* GetAddress() const;
```

Returns

The field connection data address.

Description

Returns the field connection data address.

10.1.36.3.2 - CSdpFieldConnectionData::GetAddressTypeId Method

Gets the address type ID.

C++

```
EAddressType GetAddressTypeId() const;
```

Returns

The field connection data CSdpParser::EAddressType.

Description

Returns the address type ID.

10.1.36.3.3 - CSdpFieldConnectionData::GetAddressTypeString Method

Gets the address type string.

C++

```
const char* GetAddressTypeString() const;
```

Returns

The address type string.

Description

Returns the address type string.

10.1.36.3.4 - CSdpFieldConnectionData::GetNbAddresses Method

Gets the number of addresses.

C++

```
int16_t GetNbAddresses() const;
```

Returns

The field connection data number of addresses.

Description

Returns the field connection data number of addresses.

10.1.36.3.5 - CSdpFieldConnectionData::GetNetworkTypeId Method

Gets the network type ID.

C++

```
ENetworkType GetNetworkTypeId() const;
```

Returns

The field connection data CSdpParser::ENetworkType.

Description

Returns the network type ID.

10.1.36.3.6 - CSdpFieldConnectionData::GetNetworkTypeString Method

Gets the network type string.

C++

```
const char* GetNetworkTypeString() const;
```

Returns

The network type string.

Description

Returns the network type string.

10.1.36.3.7 - CSdpFieldConnectionData::GetTtl Method

Gets TTL.

C++

```
int16_t GetTtl() const;
```

Returns

The field connection data ttl.

Description

Returns the field connection data ttl.

10.1.36.3.8 - CSdpFieldConnectionData::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.36.3.9 - CSdpFieldConnectionData::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 232).

10.1.36.3.10 - CSdpFieldConnectionData::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.36.3.11 - CSdpFieldConnectionData::SetAddress Method

Sets the address.

C++

```
void SetAddress(IN const char* pszAddress);
```

Parameters

Parameters	Description
`IN const char* pszAddress`	The address string to set.

Description

Sets the address string.

10.1.36.3.12 - CSdpFieldConnectionData::SetAddressTypeId Method

Sets the address type ID.

C++

```
void SetAddressTypeId(IN EAddressType eAddressType);
```

Parameters

Parameters	Description
`IN EAddressType eAddressType`	The EAddressType to set.

Description

Sets the address type ID.

10.1.36.3.13 - CSdpFieldConnectionData::SetAddressTypeString Method

Sets the address type string.

C++

```
void SetAddressTypeString(IN const char* pszAddressType);
```

Parameters

Parameters	Description
`eAddressType`	The address type string to set.

Description

Sets the address type string.

10.1.36.3.14 - CSdpFieldConnectionData::SetNbAddresses Method

Sets the number of addresses.

C++

```
void SetNbAddresses(IN int16_t nNbAddresses);
```

Parameters

Parameters	Description
`IN int16_t nNbAddresses`	The number of addresses to set.

Description

Sets the number of addresses.

10.1.36.3.15 - CSdpFieldConnectionData::SetNetworkTypeId Method

Sets the network type ID.

C++

```
void SetNetworkTypeId(IN ENetworkType eNetworkType);
```

Parameters

Parameters	Description
`IN ENetworkType eNetworkType`	The ENetworkType to set.

Description

Sets the network type ID.

10.1.36.3.16 - CSdpFieldConnectionData::SetNetworkTypeString Method

Sets the network type string.

C++

```
void SetNetworkTypeString(IN const char* pszNetworkType);
```

Parameters

Parameters	Description
`IN const char* pszNetworkType`	The network type string to set.

Description

Sets the network type string.

10.1.36.3.17 - CSdpFieldConnectionData::SetTtl Method

Sets TTL.

C++

```
void SetTtl(IN int16_t nTtl);
```

Parameters

Parameters	Description
`IN int16_t nTtl`	The TTL value to set.

Description

Sets the TTL value.

10.1.36.3.18 - CSdpFieldConnectionData::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.36.4 - Operators

10.1.36.4.1 - CSdpFieldConnectionData::!= Operator

Comparison operator.

C++

```
bool operator !=(IN const CSdpFieldConnectionData& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rFrom`	The CSdpFieldConnectionData (see page 228) to be compared.

Returns

true if both are not equal, false otherwise.

Description

Comparison operator.

10.1.36.4.2 - CSdpFieldConnectionData::= Operator

Assignment operator.

C++

```
CSdpFieldConnectionData& operator =(IN const CSdpFieldConnectionData& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.36.4.3 - CSdpFieldConnectionData::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldConnectionData& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rFrom`	The CSdpFieldConnectionData (see page 228) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.37 - CSdpFieldMediaAnnouncement Class

This class implements an abstraction of a media-field.

Class Hierarchy

C++

```
class CSdpFieldMediaAnnouncement : public CSdpParser;
```

Description

This class is an abstraction of a media-field in a SDP packet.

A session description may contain a number of media descriptions, each media description starts with an "m=" field. Currently defined medias are "audio", "video", "application", "data", and "control".

It follows the BNF described in RFC 2327.

RFC 2327 BNF:

```
media-field    =   "m=" media space port ["/" integer]
                   space proto 1*(space fmt) CRLF
media          =   1*(alpha-numeric)
fmt            =   1*(alpha-numeric)
proto          =   1*(alpha-numeric)
port           =   1*(DIGIT)
space          =   " "
alpha-numeric  =   ALPHA | DIGIT
```

Location

SdpParser/CSdpFieldMediaAnnouncement.h

Constructors

Constructor	Description
◆ CSdpFieldMediaAnnouncement (see page 237)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldMediaAnnouncement (see page 237)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 243)	Assignment operator.
◆ == (see page 243)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ AddMediaFormat (see page 237)	Adds a media format at the end of the list.
◆ GetMediaFormat (see page 238)	Gets the media format at the specified index.
◆ GetMediaTypeId (see page 238)	Gets the media type ID.
◆ GetMediaTypeString (see page 238)	Gets the media type string.
◆ GetNbMediaFormats (see page 238)	Gets the number of media formats.
◆ GetNbTransportPorts (see page 239)	Gets the number of tranpsport ports.
◆ GetTransportPort (see page 239)	Gets the transport port.
◆ GetTransportProtocolId (see page 239)	Gets the transport protocol ID.
◆ GetTransportProtocolString (see page 239)	Gets the transport protocol string.
◆ InsertMediaFormat (see page 239)	Inserts a media format at the specified index.
◆ Parse (see page 240)	Parses all the needed information for this field.
◆ ParseMediaFormat (see page 240)	Parses a string containing ONLY digits.
◆ RemoveMediaFormat (see page 240)	Removes the media format at the specified index.
◆ Reset (see page 241)	Resets all the data members.
◆ Serialize (see page 241)	Serializes the specified CBlob.
◆ SetMediaFormat (see page 241)	Sets the media format at the specified index.
◆ SetMediaTypeId (see page 241)	Sets the media type ID.
◆ SetMediaTypeString (see page 241)	Sets the media type string.
◆ SetNbTransportPorts (see page 242)	Sets the number of transport ports.
◆ SetTransportPort (see page 242)	Sets the transport port.
◆ SetTransportProtocolId (see page 242)	Sets the transport protocol ID.
◆ SetTransportProtocolString (see page 242)	Sets the transport protocol string.

◆ Validate (see page 243)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.37.1 - Constructors

10.1.37.1.1 - CSdpFieldMediaAnnouncement

10.1.37.1.1.1 - CSdpFieldMediaAnnouncement::CSdpFieldMediaAnnouncement Constructor

Default constructor.

C++

```
CSdpFieldMediaAnnouncement();
```

Description

Constructor

10.1.37.1.1.2 - CSdpFieldMediaAnnouncement::CSdpFieldMediaAnnouncement Constructor

Copy constructor.

C++

```
CSdpFieldMediaAnnouncement(IN const CSdpFieldMediaAnnouncement& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldMediaAnnouncement& rFrom`	The object to be copied.

Description

Copy constructor

10.1.37.2 - Destructors

10.1.37.2.1 - CSdpFieldMediaAnnouncement::~CSdpFieldMediaAnnouncement Destructor

Destructor.

C++

```
virtual ~CSdpFieldMediaAnnouncement();
```

Description

Destructor

10.1.37.3 - Methods

10.1.37.3.1 - CSdpFieldMediaAnnouncement::AddMediaFormat Method

Adds a media format at the end of the list.

C++

```
void AddMediaFormat(IN const char* pszMediaFormat);
```

Parameters

Parameters	Description
`IN const char* pszMediaFormat`	The media format to append.

Description

Adds one media format at the end of the list for the media announcement field.

10.1.37.3.2 - CSdpFieldMediaAnnouncement::GetMediaFormat Method

Gets the media format at the specified index.

C++

```
const char* GetMediaFormat(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the desired media format.

Returns

The media formats at the specified index.

Description

Returns the media format at the specified index.

10.1.37.3.3 - CSdpFieldMediaAnnouncement::GetMediaTypeId Method

Gets the media type ID.

C++

```
EMediaType GetMediaTypeId() const;
```

Returns

The field connection data CSdpParser::ENetworkType.

Description

Returns the network type ID.

10.1.37.3.4 - CSdpFieldMediaAnnouncement::GetMediaTypeString Method

Gets the media type string.

C++

```
const char* GetMediaTypeString() const;
```

Returns

The network type string.

Description

Returns the network type string.

10.1.37.3.5 - CSdpFieldMediaAnnouncement::GetNbMediaFormats Method

Gets the number of media formats.

C++

```
uint32_t GetNbMediaFormats() const;
```

Returns

The number of media formats.

Description

Returns the number of media formats.

10.1.37.3.6 - CSdpFieldMediaAnnouncement::GetNbTransportPorts Method

Gets the number of tranpsport ports.

C++

```
int16_t GetNbTransportPorts() const;
```

Returns

The number of transport port.

Description

Returns the number of transport ports.

10.1.37.3.7 - CSdpFieldMediaAnnouncement::GetTransportPort Method

Gets the transport port.

C++

```
int32_t GetTransportPort() const;
```

Returns

The transport port.

Description

Returns the value of the transport port.

10.1.37.3.8 - CSdpFieldMediaAnnouncement::GetTransportProtocolId Method

Gets the transport protocol ID.

C++

```
ETransportProtocol GetTransportProtocolId() const;
```

Returns

The transport protocol ID.

Description

Returns the transport protocol ID.

10.1.37.3.9 - CSdpFieldMediaAnnouncement::GetTransportProtocolString Method

Gets the transport protocol string.

C++

```
const char* GetTransportProtocolString() const;
```

Returns

The transport protocol string.

Description

Returns the transport protocol string.

10.1.37.3.10 - CSdpFieldMediaAnnouncement::InsertMediaFormat Method

Inserts a media format at the specified index.

C++

```
void InsertMediaFormat(IN uint16_t uIndex, IN const char* pszMediaFormat);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Where to insert the new media format.
`IN const char* pszMediaFormat`	The media format to insert.

Description

Inserts one media format at the specified index in the list.

10.1.37.3.11 - CSdpFieldMediaAnnouncement::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.37.3.12 - CSdpFieldMediaAnnouncement::ParseMediaFormat Method

Parses a string containing ONLY digits.

C++

```
static EParserResult ParseMediaFormat(IN const char* pszMediaFormat, OUT uint32_t& ruMediaFormat);
```

Parameters

Parameters	Description
`IN const char* pszMediaFormat`	The null terminated string containing the media format.
`OUT uint32_t& ruMediaFormat`	The media format found in the string. Ignores the value if the method fails.

Returns

eOK_NULL if parsing succeeded (the string only contained digits).

eERROR otherwise.

Description

Parses a string containing ONLY digits.

10.1.37.3.13 - CSdpFieldMediaAnnouncement::RemoveMediaFormat Method

Removes the media format at the specified index.

C++

```
void RemoveMediaFormat(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the media format to remove.

Description

Removes one media format at the specified index in the media announcement field.

10.1.37.3.14 - CSdpFieldMediaAnnouncement::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 240).

10.1.37.3.15 - CSdpFieldMediaAnnouncement::Serialize Method

Serializes the specified CBlob.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.37.3.16 - CSdpFieldMediaAnnouncement::SetMediaFormat Method

Sets the media format at the specified index.

C++

```
void SetMediaFormat(IN uint16_t uIndex, IN const char* pszMediaFormat);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the media format to set.
`IN const char* pszMediaFormat`	The media format to set.

Description

Sets one media format at the specified index in the list.

10.1.37.3.17 - CSdpFieldMediaAnnouncement::SetMediaTypeId Method

Sets the media type ID.

C++

```
void SetMediaTypeId(IN EMediaType eMediaType);
```

Parameters

Parameters	Description
`IN EMediaType eMediaType`	The media type ID to set.

Description

Sets the media type ID.

10.1.37.3.18 - CSdpFieldMediaAnnouncement::SetMediaTypeString Method

Sets the media type string.

C++

```
void SetMediaTypeString(IN const char* pszMediaType);
```

Parameters

Parameters	Description
`IN const char* pszMediaType`	The media type string to set.

Description

Sets the media type string.

10.1.37.3.19 - CSdpFieldMediaAnnouncement::SetNbTransportPorts Method

Sets the number of transport ports.

C++

```
void SetNbTransportPorts(IN int16_t nNbTransportPorts);
```

Parameters

Parameters	Description
`IN int16_t nNbTransportPorts`	The number of transport ports to set.

Description

Sets the number of transport ports for the media announcement field.

10.1.37.3.20 - CSdpFieldMediaAnnouncement::SetTransportPort Method

Sets the transport port.

C++

```
void SetTransportPort(IN int32_t nTransportPort);
```

Parameters

Parameters	Description
`IN int32_t nTransportPort`	The transport port to set.

Description

Sets the transport port for the media announcement field.

10.1.37.3.21 - CSdpFieldMediaAnnouncement::SetTransportProtocolId Method

Sets the transport protocol ID.

C++

```
void SetTransportProtocolId(IN ETransportProtocol eTransportProtocol);
```

Parameters

Parameters	Description
`IN ETransportProtocol eTransportProtocol`	The transport protocol ID to set.

Description

Sets the ETransportProtocol for the media announcement field.

10.1.37.3.22 - CSdpFieldMediaAnnouncement::SetTransportProtocolString Method

Sets the transport protocol string.

C++

```
void SetTransportProtocolString(IN const char* pszTransportProtocol);
```

Parameters

Parameters	Description
`IN const char* pszTransportProtocol`	The transport protocol string to set.

Description

Sets the transport protocol string for the media announcement field.

10.1.37.3.23 - CSdpFieldMediaAnnouncement::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.37.4 - Operators

10.1.37.4.1 - CSdpFieldMediaAnnouncement::= Operator

Assignment operator.

C++

```
CSdpFieldMediaAnnouncement& operator =(IN const CSdpFieldMediaAnnouncement& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldMediaAnnouncement& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.37.4.2 - CSdpFieldMediaAnnouncement::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldMediaAnnouncement& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldMediaAnnouncement& rFrom`	The CSdpFieldMediaAnnouncement (see page 235) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.38 - CSdpFieldOrigin Class

This class implements an abstraction of an origin-field.

Class Hierarchy

CSdpParser
CSdpFieldOrigin

C++

```
class CSdpFieldOrigin : public CSdpParser;
```

Description

This class is an abstraction of an origin-field in a SDP packet.

The "o=" field gives the originator of the session (their username and the address of the user's host) plus a session ID and session version number.

If follows the BNF described in RFC 2327.

RFC 2327 BNF:

```
origin-field    =    "o=" username space
                     sess-id space sess-version space
                     nettype space addrtype space
                     addr CRLF
username        =    safe
safe            =    alpha-numeric |
                     """ | """ | "-" | "." | "/" | ":" | "?" | """ |
                     "#" | "$" | "&" | "*" | ";" | "=" | "@" | "[" |
                     "]" | "^" | "_" | "`" | "{" | "|" | "}" | "+" |
                     "~" | "
sess-id         =    1*(DIGIT)
sess-version    =    1*(DIGIT)
nettype         =    "IN"
addrtype        =    "IP4" | "IP6"
addr            =    FQDN  | unicast-address
FQDN            =    4*(alpha-numeric|"-"|".")
unicast-address =    IP4-address
IP4-address     =    b1 "." decimal-uchar "." decimal-uchar "." b4
b1              =    decimal-uchar
b4              =    decimal-uchar
alpha-numeric   =    ALPHA | DIGIT
decimal-uchar   =    DIGIT
                     | POS-DIGIT DIGIT
                     | ("1" 2*(DIGIT))
                     | ("2" ("0"|"1"|"2"|"3"|"4") DIGIT)
                     | ("2" "5" ("0"|"1"|"2"|"3"|"4"|"5"))
POS-DIGIT       =    "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
```

Location

SdpParser/CSdpFieldOrigin.h

Constructors

Constructor	Description
◆ CSdpFieldOrigin (see page 246)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldOrigin (see page 246)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 251)	Assignment operator.
◆ == (see page 251)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetAddress (see page 246)	Gets the address.
◆ GetAddressTypeId (see page 246)	Gets the address type ID.
◆ GetAddressTypeString (see page 247)	Gets the address type string.
◆ GetNetworkTypeId (see page 247)	Gets the network type ID.
◆ GetNetworkTypeString (see page 247)	Gets the network type string.
◆ GetSessionId (see page 247)	Gets the session ID.
◆ GetUsername (see page 247)	Gets the username.
◆ GetVersion (see page 248)	Gets the version.
◆ Parse (see page 248)	Parses all the needed information for this field.
◆ Reset (see page 248)	Resets all the data members.
◆ Serialize (see page 248)	Generates the data blob from the data members.
◆ SetAddress (see page 249)	Sets the address.
◆ SetAddressTypeId (see page 249)	Sets the address type ID.
◆ SetAddressTypeString (see page 249)	Sets the address type string.
◆ SetNetworkTypeId (see page 249)	Sets the network type ID.
◆ SetNetworkTypeString (see page 249)	Sets the network type string.
◆ SetSessionId (see page 250)	Sets the session ID.
◆ SetUsername (see page 250)	Sets the username.
◆ SetVersion (see page 250)	Sets the version.
◆ Validate (see page 250)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆V Reset (see page 353)	Resets the data in the parser.
◆A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.38.1 - Constructors

10.1.38.1.1 - CSdpFieldOrigin

10.1.38.1.1.1 - CSdpFieldOrigin::CSdpFieldOrigin Constructor

Default constructor.

C++

```
CSdpFieldOrigin();
```

Description

Constructor

10.1.38.1.1.2 - CSdpFieldOrigin::CSdpFieldOrigin Constructor

Copy constructor.

C++

```
CSdpFieldOrigin(IN const CSdpFieldOrigin& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldOrigin& rFrom`	The object to be copied.

Description

Copy constructor

10.1.38.2 - Destructors

10.1.38.2.1 - CSdpFieldOrigin::~CSdpFieldOrigin Destructor

Destructor.

C++

```
virtual ~CSdpFieldOrigin();
```

Description

Destructor

10.1.38.3 - Methods

10.1.38.3.1 - CSdpFieldOrigin::GetAddress Method

Gets the address.

C++

```
const char* GetAddress() const;
```

Returns

The origin field address.

Description

Returns the address.

10.1.38.3.2 - CSdpFieldOrigin::GetAddressTypeId Method

Gets the address type ID.

C++

```
EAddressType GetAddressTypeId() const;
```

Returns

The origin field address type ID.

Description

Returns the address type ID.

10.1.38.3.3 - CSdpFieldOrigin::GetAddressTypeString Method

Gets the address type string.

C++

```
const char* GetAddressTypeString() const;
```

Returns

The origin field address type string.

Description

Returns the address type string.

10.1.38.3.4 - CSdpFieldOrigin::GetNetworkTypeId Method

Gets the network type ID.

C++

```
ENetworkType GetNetworkTypeId() const;
```

Returns

The origin field network type ID.

Description

Returns the network type ID.

10.1.38.3.5 - CSdpFieldOrigin::GetNetworkTypeString Method

Gets the network type string.

C++

```
const char* GetNetworkTypeString() const;
```

Returns

The origin field network type string.

Description

Returns the network type string.

10.1.38.3.6 - CSdpFieldOrigin::GetSessionId Method

Gets the session ID.

C++

```
const char* GetSessionId() const;
```

Returns

The origin field session ID.

Description

Returns the session ID.

10.1.38.3.7 - CSdpFieldOrigin::GetUsername Method

Gets the username.

C++

```
const char* GetUsername() const;
```

Returns

The origin field username.

Description

Returns the username.

10.1.38.3.8 - CSdpFieldOrigin::GetVersion Method

Gets the version.

C++

```
const char* GetVersion() const;
```

Returns

The origin field version.

Description

Returns the version.

10.1.38.3.9 - CSdpFieldOrigin::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.38.3.10 - CSdpFieldOrigin::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 248).

10.1.38.3.11 - CSdpFieldOrigin::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.38.3.12 - CSdpFieldOrigin::SetAddress Method

Sets the address.

C++

```
void SetAddress(IN const char* pszAddress);
```

Parameters

Parameters	Description
`IN const char* pszAddress`	The address to set.

Description

Sets the address in the origin field.

10.1.38.3.13 - CSdpFieldOrigin::SetAddressTypeId Method

Sets the address type ID.

C++

```
void SetAddressTypeId(IN EAddressType eAddressType);
```

Parameters

Parameters	Description
`pszNetworkType`	The address type ID to set.

Description

Sets the address type ID in the origin field.

10.1.38.3.14 - CSdpFieldOrigin::SetAddressTypeString Method

Sets the address type string.

C++

```
void SetAddressTypeString(IN const char* pszAddressType);
```

Parameters

Parameters	Description
`pszNetworkType`	The address type string to set.

Description

Sets the address type string in the origin field.

10.1.38.3.15 - CSdpFieldOrigin::SetNetworkTypeId Method

Sets the network type ID.

C++

```
void SetNetworkTypeId(IN ENetworkType eNetworkType);
```

Parameters

Parameters	Description
`IN ENetworkType eNetworkType`	The network type ID to set.

Description

Sets the network type ID in the origin field.

10.1.38.3.16 - CSdpFieldOrigin::SetNetworkTypeString Method

Sets the network type string.

C++

```
void SetNetworkTypeString(IN const char* pszNetworkType);
```

Parameters

Parameters	Description
`IN const char* pszNetworkType`	The network type string to set.

Description

Sets the network type string in the origin field.

10.1.38.3.17 - CSdpFieldOrigin::SetSessionId Method

Sets the session ID.

C++

```
void SetSessionId(IN const char* pszSessionId);
```

Parameters

Parameters	Description
`IN const char* pszSessionId`	The session ID to set.

Description

Sets the session ID in the origin field.

10.1.38.3.18 - CSdpFieldOrigin::SetUsername Method

Sets the username.

C++

```
void SetUsername(IN const char* pszUsername);
```

Parameters

Parameters	Description
`IN const char* pszUsername`	The username to set.

Description

Sets the username in the origin field.

10.1.38.3.19 - CSdpFieldOrigin::SetVersion Method

Sets the version.

C++

```
void SetVersion(IN const char* pszVersion);
```

Parameters

Parameters	Description
`IN const char* pszVersion`	The version to set.

Description

Sets the version in the origin field.

10.1.38.3.20 - CSdpFieldOrigin::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.

- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.38.4 - Operators

10.1.38.4.1 - CSdpFieldOrigin::= Operator

Assignment operator.

C++

```
CSdpFieldOrigin& operator =(IN const CSdpFieldOrigin& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldOrigin& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.38.4.2 - CSdpFieldOrigin::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldOrigin& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldOrigin& rFrom`	The CSdpFieldOrigin (see page 243) to be compared.

Returns

true if both are equal, false otherwise.

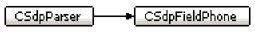
Description

Comparison operator.

10.1.39 - CSdpFieldPhone Class

This class implements an abstraction of a session-name-field.

Class Hierarchy



C++

```
class CSdpFieldPhone : public CSdpParser;
```

Description

This class is an abstraction of a session-name-field in a SDP packet.

The "p=" field is the phone number. It follows the BNF that is described in RFC 2327 and is an optional item.

RFC 2327 BNF:

```
phone-fields =      *("p=" phone-number CRLF)
phone-number =      phone | phone "(" email-safe ")" |
                    email-safe "<" phone ">"
phone        =      "+" POS-DIGIT 1*(space | "-" | DIGIT)
                    ;there must be a space or hyphen between the
                    ;international code and the rest of the number.
```

```
email-safe   =      safe | space | tab
safe         =      alpha-numeric |
                    """ | """ | "-" | "." | "/" | ":" | "?" | """ |
                    "#" | "$" | "&" | "*" | ";" | "=" | "@" | "[" |
                    "]" | "^" | "_" | "`" | "{" | "|" | "}" | "+" |
                    "~" | "
alpha-numeric =     ALPHA | DIGIT
ALPHA        =      "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|
                    "l"|"m"|"n"|"o "|"p"|"q"|"r"|"s"|"t"|"u"|"v"|
                    "w"|"x"|"y"|"z"|"A"|"B"|"C "|"D"|"E"|"F"|"G"|
                    "H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|" Q"|"R"|
                    "S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"
DIGIT        =      "0" | POS-DIGIT
POS-DIGIT    =      "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
```

Location

SdpParser/CSdpFieldPhone.h

Constructors

Constructor	Description
◆ CSdpFieldPhone (see page 253)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldPhone (see page 253)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 255)	Assignment operator.
◆ == (see page 255)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetPhone (see page 253)	Gets the phone number.
◆ Parse (see page 254)	Parses all the needed information for this field.
◆ Reset (see page 254)	Resets all the data members.
◆ Serialize (see page 254)	Generates the data blob from the data members.
◆ SetPhone (see page 254)	Sets the phone number.
◆ Validate (see page 255)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.39.1 - Constructors

10.1.39.1.1 - CSdpFieldPhone

10.1.39.1.1.1 - CSdpFieldPhone::CSdpFieldPhone Constructor

Default constructor.

C++

```
CSdpFieldPhone();
```

Description

Constructor

10.1.39.1.1.2 - CSdpFieldPhone::CSdpFieldPhone Constructor

Copy constructor.

C++

```
CSdpFieldPhone(IN const CSdpFieldPhone& from);
```

Parameters

Parameters	Description
rFrom	The object to be copied.

Description

Copy constructor

10.1.39.2 - Destructors

10.1.39.2.1 - CSdpFieldPhone::~CSdpFieldPhone Destructor

Destructor.

C++

```
virtual ~CSdpFieldPhone();
```

Description

Destructor

10.1.39.3 - Methods

10.1.39.3.1 - CSdpFieldPhone::GetPhone Method

Gets the phone number.

C++

```
const char* GetPhone() const;
```

Returns

The phone number.

Description

Returns the phone number (i.e. the p= field).

10.1.39.3.2 - CSdpFieldPhone::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

CSdpParser::EParserResult Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.39.3.3 - CSdpFieldPhone::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 254).

10.1.39.3.4 - CSdpFieldPhone::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.39.3.5 - CSdpFieldPhone::SetPhone Method

Sets the phone number.

C++

```
void SetPhone(IN const char* szPhone);
```

Parameters

Parameters	Description
`IN const char* szPhone`	The phone number.

Description

Sets the phone number (i.e. the p= field).

10.1.39.3.6 - CSdpFieldPhone::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.39.4 - Operators

10.1.39.4.1 - CSdpFieldPhone::= Operator

Assignment operator.

C++

```
CSdpFieldPhone& operator =(IN const CSdpFieldPhone& from);
```

Parameters

Parameters	Description
`rFrom`	The right operand of the assignment (to copy in *this).

Returns

CSdpFieldPhone (see page 251)& A reference to this, to enable concatenation.

Description

Assignment operator

10.1.39.4.2 - CSdpFieldPhone::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldPhone& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldPhone& rFrom`	The CSdpFieldPhone (see page 251) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.40 - CSdpFieldProtocolVersion Class

This class implements an abstraction of a proto-version field.

Class Hierarchy

CSdpParser
CSdpFieldProtocolVersion

C++

```
class CSdpFieldProtocolVersion : public CSdpParser;
```

Description

This class is an abstraction of a proto-version field in a SDP packet.

The "v=" field gives the version of the Session Description Protocol. It follows what is described in the BNF of RFC 2327.

RFC 2327 BNF:

proto-version =   "v=" 1*DIGIT CRLF

Location

SdpParser/CSdpFieldProtocolVersion.h

Constructors

Constructor	Description
◆ CSdpFieldProtocolVersion (see page 257)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldProtocolVersion (see page 257)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 259)	Assignment operator.
◆ == (see page 259)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetProtocolVersion (see page 257)	Gets the protocol version.
◆ Parse (see page 258)	Parses all the needed information for this field.
◆ Reset (see page 258)	Resets all the data members.
◆ Serialize (see page 258)	Generates the data blob from the data members.
◆ SetProtocolVersion (see page 258)	Sets the protocol version.
◆ Validate (see page 259)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.40.1 - Constructors

10.1.40.1.1 - CSdpFieldProtocolVersion

10.1.40.1.1.1 - CSdpFieldProtocolVersion::CSdpFieldProtocolVersion Constructor

Default constructor.

C++

```
CSdpFieldProtocolVersion();
```

Description

Constructor

10.1.40.1.1.2 - CSdpFieldProtocolVersion::CSdpFieldProtocolVersion Constructor

Copy constructor.

C++

```
CSdpFieldProtocolVersion(IN const CSdpFieldProtocolVersion& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldProtocolVersion& rFrom`	The object to be copied.

Description

Copy constructor

10.1.40.2 - Destructors

10.1.40.2.1 - CSdpFieldProtocolVersion::~CSdpFieldProtocolVersion Destructor

Destructor.

C++

```
virtual ~CSdpFieldProtocolVersion();
```

Description

Destructor

10.1.40.3 - Methods

10.1.40.3.1 - CSdpFieldProtocolVersion::GetProtocolVersion Method

Gets the protocol version.

C++

```
int16_t GetProtocolVersion() const;
```

Returns

The proto-version.

Description

Returns the proto-version (i.e. the v= field).

10.1.40.3.2 - CSdpFieldProtocolVersion::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.40.3.3 - CSdpFieldProtocolVersion::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 258).

10.1.40.3.4 - CSdpFieldProtocolVersion::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.40.3.5 - CSdpFieldProtocolVersion::SetProtocolVersion Method

Sets the protocol version.

C++

```
void SetProtocolVersion(IN int16_t nProtocolVersion);
```

Parameters

Parameters	Description
`IN int16_t nProtocolVersion`	The proto-version.

Description

Sets the proto-version (i.e. the v= field).

10.1.40.3.6 - CSdpFieldProtocolVersion::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.40.4 - Operators

10.1.40.4.1 - CSdpFieldProtocolVersion::= Operator

Assignment operator.

C++

```
CSdpFieldProtocolVersion& operator =(IN const CSdpFieldProtocolVersion& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldProtocolVersion& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.40.4.2 - CSdpFieldProtocolVersion::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldProtocolVersion& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldProtocolVersion& rFrom`	The CSdpFieldProtocolVersion (see page 255) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.41 - CSdpFieldSessionName Class

This class implements an abstraction of a session-name-field.

Class Hierarchy

CSdpParser
CSdpFieldSessionName

C++

```
class CSdpFieldSessionName : public CSdpParser;
```

Description

This class is an abstraction of a session-name-field in a SDP packet.

The "s=" field is the session name. It follows the BNF that is described in RFC 2327.

RFC 2327 BNF:

```
session-name-field  =  "s=" text CRLF
text                =   byte-string
byte-string         =   1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
```

Location

SdpParser/CSdpFieldSessionName.h

Constructors

Constructor	Description
◆ CSdpFieldSessionName (see page 261)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFieldSessionName (see page 261)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 263)	Assignment operator.
◆ == (see page 263)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetSessionName (see page 261)	Gets the session name.
◆ Parse (see page 262)	Parses all the needed information for this field.
◆ Reset (see page 262)	Resets all the data members.
◆ Serialize (see page 262)	Generates the data blob from the data members.

◆ SetSessionName (see page 262)	Sets the session name.
◆ Validate (see page 263)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.41.1 - Constructors

10.1.41.1.1 - CSdpFieldSessionName

10.1.41.1.1.1 - CSdpFieldSessionName::CSdpFieldSessionName Constructor

Default constructor.

C++

```
CSdpFieldSessionName();
```

Description

Constructor

10.1.41.1.1.2 - CSdpFieldSessionName::CSdpFieldSessionName Constructor

Copy constructor.

C++

```
CSdpFieldSessionName(IN const CSdpFieldSessionName& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldSessionName& rFrom`	The object to be copied.

Description

Copy constructor

10.1.41.2 - Destructors

10.1.41.2.1 - CSdpFieldSessionName::~CSdpFieldSessionName Destructor

Destructor.

C++

```
virtual ~CSdpFieldSessionName();
```

Description

Destructor

10.1.41.3 - Methods

10.1.41.3.1 - CSdpFieldSessionName::GetSessionName Method

Gets the session name.

C++

```
const char* GetSessionName() const;
```

Returns

The session-name-field.

Description

Returns the session-name-field (i.e. the s= field).

10.1.41.3.2 - CSdpFieldSessionName::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.41.3.3 - CSdpFieldSessionName::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 262).

10.1.41.3.4 - CSdpFieldSessionName::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generate the data blob from the data members.

10.1.41.3.5 - CSdpFieldSessionName::SetSessionName Method

Sets the session name.

C++

```
void SetSessionName(IN const char* pszSessionName);
```

Parameters

Parameters	Description
`IN const char* pszSessionName`	The session-name-field.

Description

Sets the session-name-field (i.e. the s= field).

10.1.41.3.6 - CSdpFieldSessionName::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.41.4 - Operators

10.1.41.4.1 - CSdpFieldSessionName::= Operator

Assignment operator.

C++

```
CSdpFieldSessionName& operator =(IN const CSdpFieldSessionName& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldSessionName& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.41.4.2 - CSdpFieldSessionName::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpFieldSessionName& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldSessionName& rFrom`	The CSdpFieldSessionName (see page 259) to be compared.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.42 - CSdpFieldTime Class

This class implements an abstraction of the time-fields.

Class Hierarchy

$$\begin{tikzcd}\boxed{\text{CSdpParser}} \arrow[r] & \boxed{\text{CSdpFieldTime}}\end{tikzcd}$$

C++

```
class CSdpFieldTime : public CSdpParser;
```

Description

This class is an abstraction of the time-fields in a SDP packet. It follows in part the BNF described in RFC 2327.

"t=" fields specify the start and stop times for a conference session.

RFC 2327 BNF:

```
time-fields        =    1*( "t=" start-time space stop-time
                        *(CRLF repeat-fields) CRLF)
                        [zone-adjustments CRLF]
repeat-fields      =    "r=" repeat-interval space typed-time
                        1*(space typed-time)
zone-adjustments   =    time space ["-"] typed-time
                        *(space time space ["-"] typed-time)
start-time         =    time | "0"
stop-time          =    time | "0"
time               =    POS-DIGIT 9*(DIGIT)
repeat-interval    =    typed-time
typed-time         =    1*(DIGIT) [fixed-len-time-unit]
fixed-len-time-unit =   "d" | "h" | "m" | "s"
space              =    " "
POS-DIGIT          =    "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
```

Location

SdpParser/CSdpFieldTime.h

Constructors

Constructor	Description
◆ CSdpFieldTime (see page 265)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFieldTime (see page 266)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 269)	Assignment operator.
◆ == (see page 269)	Comparison Operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetRepeatTime (see page 266)	Gets the vector of Repeat Time field, i.e. "r=" field.
◆ GetStartTime (see page 266)	Gets the start time.
◆ GetStopTime (see page 266)	Gets the stop time.
◆ GetTimeZone (see page 267)	Gets the Time zone field, i.e. "z=" field.
◆ Parse (see page 267)	Parses all the needed information for this field.
◆ Reset (see page 267)	Resets all the data members.
◆ Serialize (see page 267)	Generates the data blob from the data members.
◆ SetStartTime (see page 268)	Sets the start time.
◆ SetStopTime (see page 268)	Sets the stop time.
◆ SetTimeZone (see page 268)	Sets the Time zone field, i.e. "z=" field.
◆ Validate (see page 268)	Checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.42.1 - Constructors

10.1.42.1.1 - CSdpFieldTime

10.1.42.1.1.1 - CSdpFieldTime::CSdpFieldTime Constructor

Default constructor.

C++

```
CSdpFieldTime();
```

Description

Constructor

10.1.42.1.1.2 - CSdpFieldTime::CSdpFieldTime Constructor

Copy constructor.

C++

```
CSdpFieldTime(IN const CSdpFieldTime& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldTime& rFrom`	The object to be copied.

Description

Copy constructor

10.1.42.2 - Destructors

10.1.42.2.1 - CSdpFieldTime::~CSdpFieldTime Destructor

Destructor.

C++

```
virtual ~CSdpFieldTime();
```

Description

Destructor

10.1.42.3 - Methods

10.1.42.3.1 - GetRepeatTime

10.1.42.3.1.1 - CSdpFieldTime::GetRepeatTime Method

Gets the vector of Repeat Time field, i.e. "r=" field.

C++

```
CVector<CString>& GetRepeatTime();
const CVector<CString>& GetRepeatTime() const;
```

Returns

The repeat-fields value.

Description

Returns the repeat-fields value in the r= field.

10.1.42.3.2 - CSdpFieldTime::GetStartTime Method

Gets the start time.

C++

```
const char* GetStartTime() const;
```

Returns

The start-time value.

Description

Returns the start-time value in the t= field.

10.1.42.3.3 - CSdpFieldTime::GetStopTime Method

Gets the stop time.

C++

```
const char* GetStopTime() const;
```

Returns

The stop-time value.

Description

Returns the stop-time value in the t= field.

10.1.42.3.4 - CSdpFieldTime::GetTimeZone Method

Gets the Time zone field, i.e. "z=" field.

C++

```
const CString& GetTimeZone() const;
```

Returns

The timezone value.

Description

Returns the timezone value in the z= field.

10.1.42.3.5 - CSdpFieldTime::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.42.3.6 - CSdpFieldTime::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 267).

10.1.42.3.7 - CSdpFieldTime::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

draft-ietf-mmusic-sdp-new-24.txt ABNF:

```
time-fields =       1*( "t=" start-time SP stop-time
                  *(CRLF repeat-fields) CRLF)
                  [zone-adjustments CRLF]

repeat-fields =     "r=" repeat-interval SP typed-time
```

```
                 1*(SP typed-time)

zone-adjustments =    "z=" time SP ["-"] typed-time
                 *(SP time SP ["-"] typed-time)
```

10.1.42.3.8 - CSdpFieldTime::SetStartTime Method

Sets the start time.

C++

```
void SetStartTime(IN const char* pszStartTime);
```

Parameters

Parameters	Description
`IN const char* pszStartTime`	The start-time value.

Description

Sets the start-time value in the t= field.

10.1.42.3.9 - CSdpFieldTime::SetStopTime Method

Sets the stop time.

C++

```
void SetStopTime(IN const char* pszStopTime);
```

Parameters

Parameters	Description
`IN const char* pszStopTime`	The stop-time value.

Description

Sets the stop-time value in the t= field.

10.1.42.3.10 - CSdpFieldTime::SetTimeZone Method

Sets the Time zone field, i.e. "z=" field.

C++

```
void SetTimeZone(IN const char* szTimeZone);
```

Parameters

Parameters	Description
`IN const char* szTimeZone`	The timezone value.

Description

Sets the timezone value in the z= field.

10.1.42.3.11 - CSdpFieldTime::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.42.4 - Operators

10.1.42.4.1 - CSdpFieldTime::= Operator

Assignment operator.

C++

```
CSdpFieldTime& operator =(IN const CSdpFieldTime& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFieldTime& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.42.4.2 - CSdpFieldTime::== Operator

Comparison Operator.

C++

```
bool operator ==(IN const CSdpFieldTime& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpFieldTime& rFrom`	The CSdpFieldTime (see page 263) to be compared.

Returns

true if both are equal, false otherwise.

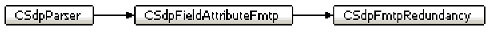
Description

Comparison operator.

10.1.43 - CSdpFmtpRedundancy Class

This class implements an abstraction of a redundancy fmtp attribute.

Class Hierarchy



C++

```
class CSdpFmtpRedundancy : public CSdpFieldAttributeFmtp;
```

Description

This class is an abstraction of a redundancy fmtp (RFC 2198) attribute in a SDP packet.

The format of this fmtp attribute is based on the format of CSdpFieldAttributeFmtp (see page 95), but is more specific in its format-specific-parameters.

Derived from RFC 2198 and RFC 2327:

```
redundancy-fmtp-attribute  = "fmtp:" format redundancy-parameters
format                     = token
redundancy-parameters      = format *( "/" format )
```

Location

SdpParser/CSdpFmtpRedundancy.h

See Also

CSdpFieldAttributeFmtp (see page 95)

Constructors

Constructor	Description
◆ CSdpFmtpRedundancy (see page 271)	Default destructor.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆ CSdpFieldAttributeFmtp (see page 97)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpFmtpRedundancy (see page 272)	Destructor.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆ V ~CSdpFieldAttributeFmtp (see page 97)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 274)	Assignment operator.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆ = (see page 100)	Assignment Operator.
◆ == (see page 101)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ V GenerateCopy (see page 272)	Generates a copy of the current object.
◆ GetRedundancyMediaFormats (see page 272)	Returns the media formats specified in the fmtp field attribute.
◆ V GetValue (see page 272)	Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.
◆ MergeRedundancyFmtp (see page 273)	Sets this redundancy fmtp attribute to have the media format contained in both source fmtp attributes in the order found in rFrom1.
◆ V Parse (see page 273)	Parses the data. Can return any type of EParserResult.
◆ V Reset (see page 273)	Resets the data in the parser.
◆ V Validate (see page 273)	Validates and checks the validity of the parsed data.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆ V GenerateCopy (see page 97)	Generates a copy of the current object.
◆ GetFmtpType (see page 98)	Gives the Fmtp type for which the format-specific-parameters pattern was implemented by the object. For the CSdpFieldAttributeFmtp (see page 95), eFMTP_TYPE_UNKNOWN is returned.
◆ GetFormat (see page 98)	Returns the media format of the Fmtp field attribute as a string.
◆ GetMediaFormat (see page 98)	Returns the media format of the Fmtp field attribute.
◆ V GetValue (see page 98)	Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.
◆ V Parse (see page 98)	Parses the data. Can return any type of EParserResult.
◆ V Reset (see page 99)	Resets the data in the parser.
◆ Serialize (see page 99)	Appends this object into the blob. Also adds a CRLF.
◆ SetFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆ SetMediaFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆ V SetValue (see page 100)	Sets the value of the Fmtp field attribute to the string.
◆ V Validate (see page 100)	Validates and checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

Enumerations

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
EFmtpType (see page 101)	

10.1.43.1 - Constructors

10.1.43.1.1 - CSdpFmtpRedundancy

10.1.43.1.1.1 - CSdpFmtpRedundancy::CSdpFmtpRedundancy Constructor

Default destructor.

C++

```
CSdpFmtpRedundancy();
```

Description

Default constructor.

10.1.43.1.1.2 - CSdpFmtpRedundancy::CSdpFmtpRedundancy Constructor

Copy constructor.

C++

```
CSdpFmtpRedundancy(IN const CSdpFmtpRedundancy& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFmtpRedundancy& rFrom`	The object to be copied.

Description

Copy constructor.

10.1.43.2 - Destructors

10.1.43.2.1 - CSdpFmtpRedundancy::~CSdpFmtpRedundancy Destructor

Destructor.

C++

```
virtual ~CSdpFmtpRedundancy();
```

Description

Destructor.

10.1.43.3 - Methods

10.1.43.3.1 - CSdpFmtpRedundancy::GenerateCopy Method

Generates a copy of the current object.

C++

```
virtual GO CSdpFieldAttributeFmtp* GenerateCopy() const;
```

Returns

A copy of the current object.

Description

Returns a copy of the current object.

Warning

Gives ownership of the new object.

10.1.43.3.2 - GetRedundancyMediaFormats

10.1.43.3.2.1 - CSdpFmtpRedundancy::GetRedundancyMediaFormats Method

Returns the media formats specified in the fmtp field attribute.

C++

```
CVector<CString>& GetRedundancyMediaFormats();
const CVector<CString>& GetRedundancyMediaFormats() const;
```

Returns

The media formats.

Description

Returns the media formats specified in the fmtp field attribute.

10.1.43.3.3 - CSdpFmtpRedundancy::GetValue Method

Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.

C++

```
virtual const char* GetValue() const;
```

Returns

The format-specific-parameters of the redundancy fmtp field attribute.

Description

Serializes the format-specific-parameters of the redundancy fmtp field attribute and returns the value.

10.1.43.3.4 - CSdpFmtpRedundancy::MergeRedundancyFmtp Method

Sets this redundancy fmtp attribute to have the media format contained in both source fmtp attributes in the order found in rFrom1.

C++

```
bool MergeRedundancyFmtp(IN const CSdpFmtpRedundancy& rFrom1, IN const CSdpFmtpRedundancy& rFrom2);
```

Parameters

Parameters	Description
`IN const CSdpFmtpRedundancy& rFrom1`	One fmtp redundancy fmtp attribute to merge with the other.
`IN const CSdpFmtpRedundancy& rFrom2`	One fmtp redundancy fmtp attribute to merge with the other.

Returns

false: One of the redundancy fmtp attributes was invalid. The object is invalid in this case.

true: The merge was successfully done. The object could still be invalid if no media format was common to both fmtp attributes.

Description

This method permits to merge redundancy capabilities.

The result of the merge is in this object. Only media formats supported in both redundancy fmtps are in the result. The media formats are in the order found in rFrom1.

Note that if no media format is present in both redundancy fmtp attributes, none is in this one and it is NOT valid.

Also note that if one of the redundancy fmtp attributes is invalid, the target attribute (this) has no media format and is NOT valid.

10.1.43.3.5 - CSdpFmtpRedundancy::Parse Method

Parses the data. Can return any type of EParserResult.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed. The pointer is advanced after the parsed data. rError Result value.

Description

Parses all the needed information for this field. An error is signaled in 'rError' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.43.3.6 - CSdpFmtpRedundancy::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 273).

10.1.43.3.7 - CSdpFmtpRedundancy::Validate Method

Validates and checks the validity of the parsed data.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.43.4 - Operators

10.1.43.4.1 - CSdpFmtpRedundancy::= Operator

Assignment operator.

C++

```
CSdpFmtpRedundancy& operator =(IN const CSdpFmtpRedundancy& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFmtpRedundancy& rFrom`	The object to copy into this one.

Returns

This object.

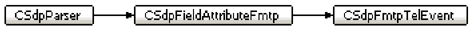
Description

Assignment operator. Copy the passed object into this one.

10.1.44 - CSdpFmtpTelEvent Class

This class implements an abstraction of a telephone-event fmtp attribute.

Class Hierarchy



C++

```
class CSdpFmtpTelEvent : public CSdpFieldAttributeFmtp;
```

Description

This class is an abstraction of a telephone-event fmtp attribute in a SDP packet as per draft-ietf-avt-rfc2833bis-09.

The format of this fmtp attribute is based on the format of CSdpFieldAttributeFmtp (see page 95), but is more specific in its format-specific-parameters.

```
tel-event-fmtp-attribute    = "fmtp:" format tel-event-parameters
format                      = token
tel-event-parameters        = tel-event-param *( "," tel-event-param )
tel-event-param             = tel-event [ "-" tel-event ]
tel-event                   = 1*DIGIT
```

Location

SdpParser/CSdpFmtpTelEvent.h

See Also

CSdpFieldAttributeFmtp (see page 95)

Constructors

Constructor	Description
CSdpFmtpTelEvent (see page 276)	Default destructor.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
CSdpFieldAttributeFmtp (see page 97)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆V ~CSdpFmtpTelEvent (see page 277)	Destructor.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆V ~CSdpFieldAttributeFmtp (see page 97)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 279)	Assignment operator.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆ = (see page 100)	Assignment Operator.
◆ == (see page 101)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆V GenerateCopy (see page 277)	Generates a copy of the current object.
◆V GetValue (see page 277)	Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.
◆ IsAnyTelephoneEventSupported (see page 277)	Verifies if any telephone events are supported.
◆ IsTelephoneEventSupported (see page 277)	Verifies if the given telephone event is supported.
◆ MergeTelEventFmtp (see page 278)	Merges the telephone-events supported by both parties into this one.
◆V Parse (see page 278)	Parses the data. Can return any type of EParserResult.
◆V Reset (see page 278)	Resets the data in the parser.
◆ SetTelEventSupport (see page 279)	Helper that sets support for a group of telephone events.
◆V Validate (see page 279)	Validates and checks the validity of the parsed data.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
◆V GenerateCopy (see page 97)	Generates a copy of the current object.
◆ GetFmtpType (see page 98)	Gives the Fmtp type for which the format-specific-parameters pattern was implemented by the object. For the CSdpFieldAttributeFmtp (see page 95), eFMTP_TYPE_UNKNOWN is returned.
◆ GetFormat (see page 98)	Returns the media format of the Fmtp field attribute as a string.
◆ GetMediaFormat (see page 98)	Returns the media format of the Fmtp field attribute.

◆ V GetValue (see page 98)	Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.
◆ V Parse (see page 98)	Parses the data. Can return any type of EParserResult.
◆ V Reset (see page 99)	Resets the data in the parser.
◆ Serialize (see page 99)	Appends this object into the blob. Also adds a CRLF.
◆ SetFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆ SetMediaFormat (see page 99)	Sets the media format of the Fmtp field attribute.
◆ V SetValue (see page 100)	Sets the value of the Fmtp field attribute to the string.
◆ V Validate (see page 100)	Validates and checks the validity of the parsed data.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
V	virtual
A	abstract

Enumerations

Enumeration	Description
ETelEventGroup (see page 280)	Telephone event control.

CSdpFieldAttributeFmtp Class

CSdpFieldAttributeFmtp Class	Description
EFmtpType (see page 101)	

10.1.44.1 - Constructors

10.1.44.1.1 - CSdpFmtpTelEvent

10.1.44.1.1.1 - CSdpFmtpTelEvent::CSdpFmtpTelEvent Constructor

Default destructor.

C++

```
CSdpFmtpTelEvent();
```

Description

Default constructor.

10.1.44.1.1.2 - CSdpFmtpTelEvent::CSdpFmtpTelEvent Constructor

Copy constructor.

C++

```
CSdpFmtpTelEvent(IN const CSdpFmtpTelEvent& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFmtpTelEvent& rFrom`	The object to be copied.

Description

Copy constructor.

10.1.44.2 - Destructors

10.1.44.2.1 - CSdpFmtpTelEvent::~CSdpFmtpTelEvent Destructor

Destructor.

C++

```
virtual ~CSdpFmtpTelEvent();
```

Description

Destructor.

10.1.44.3 - Methods

10.1.44.3.1 - CSdpFmtpTelEvent::GenerateCopy Method

Generates a copy of the current object.

C++

```
virtual GO CSdpFieldAttributeFmtp* GenerateCopy() const;
```

Returns

A copy of the current object.

Description

Returns a copy of the current object.

Warning

Gives ownership of the new object.

10.1.44.3.2 - CSdpFmtpTelEvent::GetValue Method

Serializes the Fmtp field value in m_strValue and returns its value. Also used by Serialize (see page 99) to add the format-specific-parameters to the blob.

C++

```
virtual const char* GetValue() const;
```

Returns

The format-specific-parameters of the telephone-event fmtp field attribute.

Description

Serializes the format-specific-parameters of the telephone-event fmtp field attribute and returns the value.

10.1.44.3.3 - CSdpFmtpTelEvent::IsAnyTelephoneEventSupported Method

Verifies if any telephone events are supported.

C++

```
bool IsAnyTelephoneEventSupported() const;
```

Returns

true if at least one telephone event is marked as supported in the fmtp attribute.

false otherwise.

Description

Says if at least one telephone event is supported by the fmtp field.

10.1.44.3.4 - CSdpFmtpTelEvent::IsTelephoneEventSupported Method

Verifies if the given telephone event is supported.

C++

```
bool IsTelephoneEventSupported(IN uint8_t uTelEvent) const;
```

Returns

true if the event is present in the fmtp attribute.

false otherwise.

Description

Sees if the event is present in the fmtp attribute.

10.1.44.3.5 - CSdpFmtpTelEvent::MergeTelEventFmtp Method

Merges the telephone-events supported by both parties into this one.

C++

```
bool MergeTelEventFmtp(IN const CSdpFmtpTelEvent& rFrom1, IN const CSdpFmtpTelEvent& rFrom2);
```

Parameters

Parameters	Description
`IN const CSdpFmtpTelEvent& rFrom1`	One fmtp telephone-event fmtp attribute to merge with the other.
`IN const CSdpFmtpTelEvent& rFrom2`	One fmtp telephone-event fmtp attribute to merge with the other.

Returns

false: One of the telephone-event fmtp attributes was invalid. The object is invalid in this case.

true: The merge was successfully done. The object could still be invalid if no event was common to both fmtp attributes.

Description

This method permits to merge telephone-event capabilities.

The result of the merge is in this object. Only events supported in both telephone-event fmtps are in the result.

Note that if no event is present in both telephone-event fmtp attributes, none is in this one and it is NOT valid.

Also note that if one of the telephone-event fmtp attributes is invalid, the target attribute (this) has no event and is NOT valid.

10.1.44.3.6 - CSdpFmtpTelEvent::Parse Method

Parses the data. Can return any type of EParserResult.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed. The pointer is advanced after the parsed data. rres Result value.

Description

Parses all the needed information for this field. An error is signaled in 'rres' if the data couldn't be parsed or if an EOL wasn't found at the end of the data.

10.1.44.3.7 - CSdpFmtpTelEvent::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 278).

10.1.44.3.8 - SetTelEventSupport

10.1.44.3.8.1 - CSdpFmtpTelEvent::SetTelEventSupport Method

Helper that sets support for a group of telephone events.

C++

```
void SetTelEventSupport(IN ETelEventGroup eGroup, IN bool bEnable);
```

Parameters

Parameters	Description
`IN ETelEventGroup eGroup`	Group of events to enable/disable. See definition of ETelEventGroup (see page 280) for details.
`IN bool bEnable`	Set to true to enable event.

Description

Helper that sets support for a group of telephone events.

See Also

IsTelEventSupportedByLocalConfig

10.1.44.3.8.2 - CSdpFmtpTelEvent::SetTelEventSupport Method

Sets support for a single telephone event.

C++

```
void SetTelEventSupport(IN uint8_t uEvent, IN bool bEnable);
```

Parameters

Parameters	Description
`IN uint8_t uEvent`	Number of the event, as per draft-ietf-avt-rfc2833bis-09.txt. Valid values are in the range 0-255.
`IN bool bEnable`	Set to true to enable event.

Description

Sets support for a single telephone event.

See Also

IsTelephoneEventSupported (see page 277)

10.1.44.3.9 - CSdpFmtpTelEvent::Validate Method

Validates and checks the validity of the parsed data.

C++

```
virtual bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

A telephone-event fmtp attribute is only valid when at least one event is activated.

10.1.44.4 - Operators

10.1.44.4.1 - CSdpFmtpTelEvent::= Operator

Assignment operator.

C++

```
CSdpFmtpTelEvent& operator =(IN const CSdpFmtpTelEvent& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpFmtpTelEvent& rFrom`	The object to copy into this one.

Returns

This object.

Description

Assignment operator. Copy the passed object into this one.

10.1.44.5 - Enumerations

10.1.44.5.1 - CSdpFmtpTelEvent::ETelEventGroup Enumeration

Telephone event control.

C++

```
enum ETelEventGroup {
  eDTMF,
  eFLASH,
  eBASIC_FAXMODEM
};
```

Members

Members	Description
`eDTMF`	Events 0 through 15 inclusive.
`eFLASH`	Event 16 only.
`eBASIC_FAXMODEM`	Events 32 through 36 inclusive and event 49.

10.1.45 - CSdpKeyManagementParameter Class

This class implements the base class for handling parameters related to key management attribute.

Class Hierarchy

C++

```
class CSdpKeyManagementParameter;
```

Description

This class is the base class for handling parameters related to key management attribute. It permits an application to set capabilities pertaining to the handling of key management attributes.

It is possible for a media to have key management parameters even though the key management attribute is located at the session level. This is the case for certain key management protocols (such as MIKEY) which can handle the media info even if at the session level.

Location

SdpParser/CSdpKeyManagementParameter.h

See Also

CSdpMikeyKeyManagementParameter

Constructors

Constructor	Description
◆ CSdpKeyManagementParameter (see page 281)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpKeyManagementParameter (see page 282)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ == (see page 282)	Comparison operator.

Legend

◆	Method

Methods

Method	Description
◆ V GenerateCopy (see page 282)	Generates a copy of the object.
◆ GetType (see page 282)	Gets the type of key management parameter.

Legend

◆	Method
V	virtual

10.1.45.1 - Constructors

10.1.45.1.1 - CSdpKeyManagementParameter

10.1.45.1.1.1 - CSdpKeyManagementParameter::CSdpKeyManagementParameter Constructor

Default constructor.

C++

```
CSdpKeyManagementParameter();
```

Description

Constructor.

10.1.45.1.1.2 - CSdpKeyManagementParameter::CSdpKeyManagementParameter Constructor

Constructor with specific EKeyManagementType.

C++

```
CSdpKeyManagementParameter(IN EKeyManagementType eType);
```

Parameters

Parameters	Description
`IN EKeyManagementType eType`	The type of parameter

Description

Constructor.

10.1.45.1.1.3 - CSdpKeyManagementParameter::CSdpKeyManagementParameter Constructor

Copy Constructor.

C++

```
CSdpKeyManagementParameter(IN const CSdpKeyManagementParameter& rSrc);
```

Description

Copy constructor.

10.1.45.2 - Destructors

10.1.45.2.1 - CSdpKeyManagementParameter::~CSdpKeyManagementParameter Destructor

Destructor.

C++

```
virtual ~CSdpKeyManagementParameter();
```

Description

Destructor.

10.1.45.3 - Methods

10.1.45.3.1 - CSdpKeyManagementParameter::GenerateCopy Method

Generates a copy of the object.

C++

```
virtual GO CSdpKeyManagementParameter* GenerateCopy() const;
```

Returns

A copy of this key management parameter. Ownership is given.

Description

Creates a copy of this key management parameter.

10.1.45.3.2 - CSdpKeyManagementParameter::GetType Method

Gets the type of key management parameter.

C++

```
EKeyManagementType GetType() const;
```

Returns

The type of key management parameter.

Description

Returns the type of the key management parameter.

10.1.45.4 - Operators

10.1.45.4.1 - CSdpKeyManagementParameter::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpKeyManagementParameter& rSrc) const;
```

Parameters

Parameters	Description
`IN const CSdpKeyManagementParameter& rSrc`	Parameter to compare with.

Returns

true if both are equal, false otherwise.

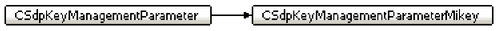
Description

Comparison operator.

10.1.46 - CSdpKeyManagementParameterMikey Class

This class is a container object used to store the media level parameters used to generate a MIKEY message.

Class Hierarchy



C++

```
class CSdpKeyManagementParameterMikey : public CSdpKeyManagementParameter;
```

Description

This class is a container object used to store the media level parameters used to generate a MIKEY message.

Location

SdpParser/CKeyMikeyManagementParameters.h

See Also

CKeyManagementParameters

Constructors

Constructor	Description
◆ CSdpKeyManagementParameterMikey (see page 284)	Default Constructor.

CSdpKeyManagementParameter Class

CSdpKeyManagementParameter Class	Description
◆ CSdpKeyManagementParameter (see page 281)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpKeyManagementParameterMikey (see page 284)	Destructor.

CSdpKeyManagementParameter Class

CSdpKeyManagementParameter Class	Description
◆ V ~CSdpKeyManagementParameter (see page 282)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 288)	Assignment Operator.
◆ == (see page 288)	Comparison operator

CSdpKeyManagementParameter Class

CSdpKeyManagementParameter Class	Description
◆ == (see page 282)	Comparison operator.

Legend

◆	Method

Methods

Method	Description
◆ V GenerateCopy (see page 285)	Generates a copy of the object.
◆ GetCryptoSession (see page 285)	Gets the crypto session for the associated stream direction.
◆ GetSecurityPolicy (see page 285)	Gets the security policy for the associated stream direction.
◆ GetSecurityPolicyCapabilities (see page 285)	Gets the security policy capabilities for the associated stream direction.
◆ GetSsrc (see page 286)	Gets the SSRC value for the associated stream direction.

◆ GetTek (see page 286)	Gets the MIKEY key for the associated stream direction.
◆ SetCryptoSession (see page 286)	Sets the crypto session for the associated stream direction.
◆ SetOutgoingSsrc (see page 287)	Sets the SSRC value for the associated stream direction.
◆ SetSecurityPolicy (see page 287)	Sets the security policy for the associated stream direction.
◆ SetSecurityPolicyCapabilities (see page 287)	Sets the security policy capabilities for the associated stream direction.

CSdpKeyManagementParameter Class

CSdpKeyManagementParameter Class	Description
◆V GenerateCopy (see page 282)	Generates a copy of the object.
◆ GetType (see page 282)	Gets the type of key management parameter.

Legend

◆	Method
V	virtual

10.1.46.1 - Constructors

10.1.46.1.1 - CSdpKeyManagementParameterMikey

10.1.46.1.1.1 - CSdpKeyManagementParameterMikey::CSdpKeyManagementParameterMikey Constructor

Default Constructor.

C++

```
CSdpKeyManagementParameterMikey();
```

Description

Constructor.

10.1.46.1.1.2 - CSdpKeyManagementParameterMikey::CSdpKeyManagementParameterMikey Constructor

Copy Constructor.

C++

```
CSdpKeyManagementParameterMikey(IN const CSdpKeyManagementParameterMikey& rSrc);
```

Parameters

Parameters	Description
`rFrom`	Source from which to copy information.

Description

Copy constructor.

10.1.46.2 - Destructors

10.1.46.2.1 - CSdpKeyManagementParameterMikey::~CSdpKeyManagementParameterMikey Destructor

Destructor.

C++

```
virtual ~CSdpKeyManagementParameterMikey();
```

Description

Destructor

10.1.46.3 - Methods

10.1.46.3.1 - CSdpKeyManagementParameterMikey::GenerateCopy Method

Generates a copy of the object.

C++

```
virtual GO CSdpKeyManagementParameter* GenerateCopy() const;
```

Returns

A copy of this object. Ownership is given.

Description

Creates a copy of this object.

10.1.46.3.2 - CSdpKeyManagementParameterMikey::GetCryptoSession Method

Gets the crypto session for the associated stream direction.

C++

```
mxt_result GetCryptoSession(IN CSdpParser::ERtpStreamDirection eDirection, OUT IMikeyCryptoSession**
ppCryptoSession) const;
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`OUT IMikeyCryptoSession** ppCryptoSession`	The crypto session to get for this stream.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: NULL pointer.

Description

Gets the IMikeyCryptoSession interface contained in this parameter for the specified stream direction. If the crypto session is not set, the IMikeyCryptoSession is set to NULL.

NOTES: It is up to the application to release the interface when it is finished using it.

10.1.46.3.3 - CSdpKeyManagementParameterMikey::GetSecurityPolicy Method

Gets the security policy for the associated stream direction.

C++

```
mxt_result GetSecurityPolicy(IN CSdpParser::ERtpStreamDirection eDirection, OUT IMikeySecurityPolicy**
ppSecurityPolicy) const;
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`OUT IMikeySecurityPolicy** ppSecurityPolicy`	The security policy to get for this stream.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: NULL pointer

Description

Gets the IMikeySecurityPolicy interface contained in this parameter for the specified stream direction. If no security policy is set, the IMikeySecurityPolicy parameter is set to NULL.

NOTES: It is up to the application to release the interface when it is finished using it.

10.1.46.3.4 - CSdpKeyManagementParameterMikey::GetSecurityPolicyCapabilities Method

Gets the security policy capabilities for the associated stream direction.

C++

```
mxt_result GetSecurityPolicyCapabilities(IN CSdpParser::ERtpStreamDirection eDirection, OUT
IMikeySecurityPolicyCapabilities** ppSecurityPolicyCapabilities) const;
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`OUT IMikeySecurityPolicyCapabilities** ppSecurityPolicyCapabilities`	The security policy capabilities to get for the particular stream.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: NULL pointer.

Description

Gets the IMikeySecurityPolicyCapabilities interface contained in this parameter for the specified stream direction. If no security policy capabilities is set, the IMikeySecurityPolicyCapabilities is set to NULL.

NOTES: It is up to the application to release the interface when it is finished using it.

10.1.46.3.5 - CSdpKeyManagementParameterMikey::GetSsrc Method

Gets the SSRC value for the associated stream direction.

C++

```
mxt_result GetSsrc(IN CSdpParser::ERtpStreamDirection eDirection, OUT uint32_t* puSsrc) const;
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`OUT uint32_t* puSsrc`	The retrieved SSRC.

Returns

-resS_OK: Operation successful. -resFE_FAIL: Operation failed. -resFE_INVALID_ARGUMENT: NULL pointer.

Description

Gets the SSRC for the desired direction. This is to be passed to RTP.

10.1.46.3.6 - CSdpKeyManagementParameterMikey::GetTek Method

Gets the MIKEY key for the associated stream direction.

C++

```
mxt_result GetTek(IN CSdpParser::ERtpStreamDirection eDirection, IN unsigned int uTekLength, OUT IMikeyKey**
ppTekKey) const;
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream for this key.
`IN unsigned int uTekLength`	The length of the key to generate.
`OUT IMikeyKey** ppTekKey`	The IMikeyKey interface contained in this attribute for the desired stream.

Returns

-resS_OK: Operation successful. -resFE_FAIL: Operation failed. -resFE_INVALID_ARGUMENT: Pointer passed is NULL. -resSI_FALSE: No more TEK keys available.

Description

Gets the TEK key contained in the desired stream. If more than one TEK is available in the crypto session, calling this method again returns the second key. If no IMikeyCryptoSession is set, the IMikeyKey is set to NULL.

NOTES: It is up to the application to release the interface when it is finished using it.

10.1.46.3.7 - CSdpKeyManagementParameterMikey::SetCryptoSession Method

Sets the crypto session for the associated stream direction.

C++

```
mxt_result SetCryptoSession(IN CSdpParser::ERtpStreamDirection eDirection, IN IMikeyCryptoSession*
pCryptoSession);
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`IN IMikeyCryptoSession* pCryptoSession`	The crypto session to set for this stream.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: NULL pointer.

Description

Sets the IMikeyCryptoSession interface contained in this parameter.

NOTES: A crypto session should never be set to multiple CSdpKeyManagementParameterMikey (see page 283) as each MUST be unique.

10.1.46.3.8 - CSdpKeyManagementParameterMikey::SetOutgoingSsrc Method

Sets the SSRC value for the associated stream direction.

C++

```
mxt_result SetOutgoingSsrc(IN uint32_t uSsrc);
```

Parameters

Parameters	Description
`IN uint32_t uSsrc`	The value to give to the SSRC for the outgoing stream.

Returns

-resS_OK: Operation successful. -resFE_FAIL: Operation failed.

Description

Sets the SSRC value of the outgoing RTP stream for this attribute that is set in MIKEY. It is not possible to set it for the incoming stream as it is controlled by the remote side.

10.1.46.3.9 - CSdpKeyManagementParameterMikey::SetSecurityPolicy Method

Sets the security policy for the associated stream direction.

C++

```
mxt_result SetSecurityPolicy(IN CSdpParser::ERtpStreamDirection eDirection, IN IMikeySecurityPolicy*
pSecurityPolicy);
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`IN IMikeySecurityPolicy* pSecurityPolicy`	The security policy to set for this stream.

Returns

-resS_OK: Operation successful. -resFE_INVALID_ARGUMENT: NULL pointer. -IMikeySecurityPolicy Set and Get error codes.

Description

Sets the IMikeySecurityPolicy for the specified stream direction.

NOTES: A security policy may be set to multiple CSdpKeyManagementParameterMikey (see page 283). If this is the case, then modifying one policy changes all the security policies that have been set with this particular IMikeySecurityPolicy. It is up to the application to make that sure that this is the desired behaviour. If not, the application MUST create a security policy for each capability and direction.

10.1.46.3.10 - CSdpKeyManagementParameterMikey::SetSecurityPolicyCapabilities Method

Sets the security policy capabilities for the associated stream direction.

C++

```
mxt_result SetSecurityPolicyCapabilities(IN CSdpParser::ERtpStreamDirection eDirection, IN
IMikeySecurityPolicyCapabilities* pSecurityPolicyCapabilities);
```

Parameters

Parameters	Description
`IN CSdpParser::ERtpStreamDirection eDirection`	The direction of the stream.
`IN IMikeySecurityPolicyCapabilities* pSecurityPolicyCapabilities`	The security policy capability to set for a particular stream.

Returns

-resS_OK: Operation successful. -resFE_FAIL: Operation failed.

Description

Sets the IMikeySecurityPolicyCapabilities interface to this parameter.

NOTES: A security policy capability may be set to multiple CSdpKeyManagementParameterMikey (see page 283). If this is the case, then modifying one capability changes all the security policies capabilities that have been set with this particular IMikeySecurityPolicyCapabilities. It is up to the application to make sure that this is the desired behaviour. If not, the application MUST create a security policy for each capability and direction.

10.1.46.4 - Operators

10.1.46.4.1 - CSdpKeyManagementParameterMikey::= Operator

Assignment Operator.

C++

```
CSdpKeyManagementParameterMikey& operator =(IN const CSdpKeyManagementParameterMikey& rSrc);
```

Parameters

Parameters	Description
`rFrom`	Source from which to copy information.

Description

Assignment operator.

10.1.46.4.2 - CSdpKeyManagementParameterMikey::== Operator

Comparison operator

C++

```
bool operator ==(IN const CSdpKeyManagementParameterMikey& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpKeyManagementParameterMikey& rFrom`	Source from which to compare information.

Returns

true if both are equal, false otherwise.

Description

Comparison operator.

10.1.47 - CSdpLevelMedia Class

This class implements an abstraction of a media-descriptions part.

Class Hierarchy

CSdpParser
CSdpLevelMedia

C++

```
class CSdpLevelMedia : public CSdpParser;
```

Description

This class is an abstraction of a media-descriptions part in a SDP packet. When the description for T.38 is used, this class is also necessary to Set and Get T.38 information. It follows the BNF that is described in RFC 2327.

RFC 2327 BNF:

```
media-descriptions =  *( media-field
                        information-field
                        *(connection-field)
                        bandwidth-fields
                        key-field
                        attribute-fields )
attribute-fields   =    *("a=" attribute CRLF)
attribute          =    attribute-rtpmap /
                        attribute-fmtp /
                        attribute-key-mgmt /
                        attribute-ptime /
                        attribute-fill-bit-removal /
                        attribute-max-bit-rate /
                        attribute-max-datagram /
                        attribute-t38-error-control /
                        attribute-t38-facsimile-max-buffer /
                        attribute-t38-Facsimile-rate-mgmnt /
                        attribute-transcoding-mmr /
                        attribute-transcoding-jbig /
                        attribute-version /
                        attribute-other
```

Fields with no link are not implemented yet, they are ignored when they are parsed.

Location

SdpParser/CSdpLevelMedia.h

See Also

CSdpFieldMediaAnnouncement (see page 235), CSdpFieldConnectionData (see page 228), CSdpFieldAttributeFmtp (see page 95), CSdpFieldAttributeKeyMgmt (see page 134), CSdpFieldAttributeRtpmap (see page 187), CSdpFieldAttributePtime (see page 177), CSdpFieldAttributeFillBitRemoval (see page 90), CSdpFieldAttributeMaxBitRate (see page 149), CSdpFieldAttributeMaxDatagram (see page 153), CSdpFieldAttributeT38ErrorControl (see page 198), CSdpFieldAttributeT38FacsimileMaxBuffer (see page 202), CSdpFieldAttributeT38FacsimileRateMgmnt (see page 206), CSdpFieldAttributeTranscodingMMR (see page 218), CSdpFieldAttributeTranscodingJBIG (see page 213), CSdpFieldAttributeVersion (see page 224), CSdpFieldAttributePreCondDes (see page 174), CSdpFieldAttributePreCondConf (see page 169), CSdpFieldAttributePreCondCurr (see page 172)

Constructors

Constructor	Description
◆ CSdpLevelMedia (see page 292)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpLevelMedia (see page 293)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 325)	Assignment operator.
◆ == (see page 325)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ AddConnectionData (see page 293)	Adds a connection data at the media level.
◆ AddCrypto (see page 293)	Adds one crypto attribute in the media.
◆ AddFmtp (see page 293)	Adds a fmtp.
◆ AddKeyMgmt (see page 293)	Adds a key management at the media level.
◆ AddKeyMgmtParam (see page 294)	Adds a key management paremeter in the media.
◆ AddOtherAttribute (see page 294)	Adds an unknown attribute in the media.
◆ AddRtpmap (see page 294)	Adds a rtp map.
◆ ClearExplicitConnectionDatas (see page 294)	Clears all connection data of the media level.
◆ FindCandidate (see page 295)	Searches for a candidate with the corresponding address and port.
◆ FindRtpMapIndexByEncodingName (see page 295)	Finds the rtpmap index for the given encoding name.
◆ GetBandwidth (see page 295)	Gets the vector of Bandwidth field, i.e. "b=" field.
◆ GetConfPreCondVector (see page 296)	Gets the vector of CONF precondition. i.e. "a=conf" field.
◆ GetConnectionData (see page 296)	Gets the connection data at the specified index.
◆ GetCrypto (see page 296)	Gets the Crypto stored at the specified index.
◆ GetCurrPreCondVector (see page 297)	Gets the vector of CURR precondition. i.e. "a=curr" field.
◆ GetDesPreCondVector (see page 297)	Gets the vector of DES precondition. i.e. "a=des" field.
◆ GetDirection (see page 297)	Gets the direction stored in the media level or in the session.
◆ GetEncodingNameFromPayloadType (see page 297)	Searches the encoding name from the payload type in the rtpmap attributes.
◆ GetEncryptionKey (see page 298)	Gets the Encryption key field, i.e. "k=" field.
◆ GetExplicitConnectionData (see page 298)	Gets the connection data at the specified index.
◆ GetExplicitDirection (see page 298)	Gets the direction stored in the media level.
◆ GetFillBitRemoval (see page 299)	Gets the fillbit removal field attribute.
◆ GetFmtp (see page 299)	Gets the Fmtp attribute at the specified index.
◆ GetFmtpFromEncoding (see page 299)	Gets the Fmtp attribute from the compression algorithm.
◆ GetFmtpFromPayloadType (see page 300)	Gets the Fmtp attribute from the payload type.
◆ GetFmtpRedundancy (see page 301)	Gets the Fmtp redundancy attribute.
◆ GetFmtpTelEvent (see page 301)	Gets the Fmtp telephone event attribute.
◆ GetIceCandidate (see page 301)	Returns a reference to the list of ICE candidate.
◆ GetIcePassword (see page 302)	Returns the ice-pwd attribute.
◆ GetIceRemoteCandidatesAttribute (see page 302)	Returns a reference to the ICE remote-candidates attribute.
◆ GetIceUserFragment (see page 302)	Returns the ice-ufrag attribute.
◆ GetInformation (see page 302)	Gets the information field, i.e. "i=" field.
◆ GetKeyMgmt (see page 303)	Gets the key management at the specified index.
◆ GetKeyMgmtParam (see page 303)	Gets the key management parameter at the specified index.
◆ GetMaxBitRate (see page 303)	Gets the maximum bitrate field attribute.
◆ GetMaxDatagram (see page 303)	Gets the maximum datagram field attribute.
◆ GetMediaAnnouncement (see page 304)	Provides access to the media announcement stored in the session.
◆ GetMid (see page 304)	Gets the mid attribute field, i.e. "a=mid:" field.
◆ GetMptimeVector (see page 304)	Gets the vector of mptime.
◆ GetNbConnectionDatas (see page 305)	Gets the number of connection data. Default value if no data.
◆ GetNbCrypto (see page 305)	Gets the number of Crypto stored in the session.

GetNbExplicitConnectionDatas (see page 305)	Provides the exact number of connection data.
GetNbFmtps (see page 306)	Gets the number of Fmtp in the media.
GetNbKeyMgmt (see page 306)	Gets the number of key management attributes.
GetNbKeyMgmtParam (see page 306)	Gets the number of key management parameters.
GetNbOtherAttributes (see page 306)	Gets the number of unknown field attributes.
GetNbParsedPtimes (see page 306)	Returns the number of parsed ptimes.
GetNbRtpmaps (see page 307)	Gets the number of rtp maps.
GetOtherAttribute (see page 307)	Gets the unknown field attribute at the specified index.
GetOtherAttributes (see page 307)	Gets the unknown field attributes.
GetPayloadTypeFromEncoding (see page 308)	Searches the payload type from the encoding name in the rtpmap attributes.
GetPtime (see page 308)	Gets the last seen ptimes in the parsing process.
GetRtpmap (see page 309)	Gets the rtp map at the specified index.
GetSdpFieldAttributeRtcp (see page 309)	Gets the Rtcp field attribute.
GetSession (see page 309)	Gets the session lvel.
GetSilenceSuppressionSupport (see page 309)	Gets the silence suppression attribute from this media.
GetT38ErrorControl (see page 310)	Gets the T38 error control field attribute.
GetT38FacsimileMaxBuffer (see page 310)	Gets the T38 Facsimile maximum buffer field attribute.
GetT38FacsimileRateMgmnt (see page 310)	Gets the T38 Facsimile rate management field attribute.
GetTranscodingJBIG (see page 311)	Gets the transcoding JBIG field attribute.
GetTranscodingMMR (see page 311)	Gets the transcoding MMR field attribute.
GetVersion (see page 311)	Gets the version field attribute.
InsertCrypto (see page 311)	Inserts the crypto attribute in the media at the specified index.
InsertRtpmap (see page 312)	Inserts a rtp map at the specified index.
IsExplicitInactive (see page 312)	Verifies whether or not the direction is set to inactive or if the session is inactive.
IsExplicitRecvOnly (see page 312)	Verifies whether or not the direction is set to receive only.
IsExplicitSendOnly (see page 312)	Verifies whether or not the direction is set to send only.
IsExplicitSendRecv (see page 312)	Verifies whether or not the direction is set to send and receive.
IsIceAttributePresent (see page 313)	Returns true if the media contains at least one ICE attribute.
IsIceMismatch (see page 313)	The attribute a=ice-mismatch is present.
IsInactive (see page 313)	Verifies whether or not the direction is set to inactive.
IsMicroLitePortPresent (see page 313)	Checks that all candidates have the microliteport extension.
IsRecvOnly (see page 314)	Verifies whether or not the direction or the session is set to receive only.
IsRtcpDeactivated (see page 314)	Checks if RTCP is disabled for this media.
IsRtcpMuxPresent (see page 314)	Returns if a rtcp-mux attribute is present.
IsSendOnly (see page 314)	Verifies whether or not the direction or the session is set to send only.
IsSendRecv (see page 315)	Verifies whether or not the direction or the session is set to send and receive.
Parse (see page 315)	Parses all the needed information for this field.
ParseKeyMgmt (see page 315)	Parses all the needed information for the key management.
RemoveCrypto (see page 315)	Removes the crypto attribute in the media at the specified index.
RemoveFmtp (see page 316)	Removes a fmtp at the specified index.
RemoveFmtpFromEncoding (see page 316)	Removes fmtp from encoding from the specified compression algorithm.
RemoveFmtpFromPayloadType (see page 317)	Removes fmtp from the specified payload type.
RemoveFmtpRedundancy (see page 317)	Removes fmtp redundancy from the media.
RemoveFmtpTelEvent (see page 317)	Removes fmtp telephone event from the media.
RemoveKeyMgmt (see page 318)	Removes the key management at the specified index.
RemoveKeyMgmtParam (see page 318)	Removes the key management parameter at the specified index.
RemoveRtpmap (see page 318)	Removes a rtp map at the specified index.
Reset (see page 318)	Resets all the data members.
Serialize (see page 318)	Generates the data blob from the data members.
SetDirection (see page 319)	Sets the direction flag.
SetEncryptionKey (see page 319)	Sets the Encryption key field, i.e. "k=" field.
SetFillBitRemoval (see page 319)	Sets the fill bit removal field attribute.
SetIceMismatch (see page 319)	Sets the a=ice-mismatch attribute.
SetInactive (see page 320)	Sets the inactive flag.
SetInformation (see page 320)	Sets the information field, i.e. "i=" field.
SetMaxBitRate (see page 320)	Sets the maximum bit rate field attribute.
SetMaxDatagram (see page 320)	Sets the maximum datagram field attribute.
SetMediaAnnouncement (see page 321)	Sets the media announcement field.
SetMicroLiteDefaultFamily (see page 321)	Specify which address family must be prioritized for the default destination when sending an offer.

◆ SetPtime (see page 321)	Sets the ptime.
◆ SetRecvOnly (see page 321)	Sets the receive only flag.
◆ SetRtcpMux (see page 321)	Returns if a rtcp-mux attribute is present.
◆ SetSdpFieldAttributeRtcp (see page 322)	Sets the rtcp field attribute.
◆ SetSendDirection (see page 322)	Sets the sending direction.
◆ SetSendOnly (see page 322)	Sets the send only flag.
◆ SetSendRecv (see page 322)	Sets the send and receive flag.
◆ SetSession (see page 323)	Sets the session level.
◆ SetT38BooleanEncoding (see page 323)	Sets the T.38 boolean encoding method for the T38FaxFillBitRemoval, T38FaxTranscodingMMR, and T38FaxTranscodingJBIG attributes.
◆ SetT38ErrorControl (see page 323)	Sets the T38 error control field attribute.
◆ SetT38FacsimileMaxBuffer (see page 323)	Sets the T38 Facsimile maximum buffer field attribute.
◆ SetT38FacsimileRateMgmnt (see page 324)	Sets the T38 Facsimile rate management field attribute.
◆ SetTranscodingJBIG (see page 324)	Sets the transcoding JBIG field attribute.
◆ SetTranscodingMMR (see page 324)	Sets the transcoding MMR field attribute.
◆ SetVersion (see page 324)	Sets the version field attribute.
◆ Validate (see page 324)	Checks the validity of the parsed data.
◆ ValidateIceCandidates (see page 325)	Validates that the IP address used by the media is also an ICE candidate.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.47.1 - Constructors

10.1.47.1.1 - CSdpLevelMedia

10.1.47.1.1.1 - CSdpLevelMedia::CSdpLevelMedia Constructor

Default constructor.

C++

```
CSdpLevelMedia();
```

Description

Constructor

10.1.47.1.1.2 - CSdpLevelMedia::CSdpLevelMedia Constructor

Copy constructor.

C++

```
CSdpLevelMedia(IN const CSdpLevelMedia& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rFrom`	The object to be copied.

Description

Copy constructor

10.1.47.2 - Destructors

10.1.47.2.1 - CSdpLevelMedia::~CSdpLevelMedia Destructor

Destructor.

C++

```
virtual ~CSdpLevelMedia();
```

Description

Destructor

10.1.47.3 - Methods

10.1.47.3.1 - CSdpLevelMedia::AddConnectionData Method

Adds a connection data at the media level.

C++

```
void AddConnectionData(IN const CSdpFieldConnectionData& rConnectionData);
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rConnectionData`	Reference to a CSdpFieldConnectionData (see page 228).

Description

Adds the connection data and calls Validate (see page 324).

10.1.47.3.2 - CSdpLevelMedia::AddCrypto Method

Adds one crypto attribute in the media.

C++

```
void AddCrypto(IN const CSdpFieldAttributeCrypto& rCrypto);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeCrypto& rCrypto`	Reference to a CSdpFieldAttributeCrypto (see page 84) to store in the session.

Description

Appends the provided Crypto in the session at the end of the vector, then calls Validate (see page 324).

10.1.47.3.3 - CSdpLevelMedia::AddFmtp Method

Adds a fmtp.

C++

```
void AddFmtp(IN const CSdpFieldAttributeFmtp& rFmtp);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeFmtp& rFmtp`	The fmtp attribute to add.

Description

Adds the given fmtp attribute to the level media.

10.1.47.3.4 - CSdpLevelMedia::AddKeyMgmt Method

Adds a key management at the media level.

C++

```
void AddKeyMgmt(IN const CSdpFieldAttributeKeyMgmt& rKeyMgmt, IN const
CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmt& rKeyMgmt`	The key management attribute to add.
`IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH`	The role that the key management attribute will use.

Description

Adds a key management attribute to the media. Note that any role previously set is overidden.

10.1.47.3.5 - CSdpLevelMedia::AddKeyMgmtParam Method

Adds a key management paremeter in the media.

C++

```
void AddKeyMgmtParam(IN const CSdpKeyManagementParameter& rKeyMgmtParam);
```

Parameters

Parameters	Description
`IN const CSdpKeyManagementParameter& rKeyMgmtParam`	The key management parameter.

Description

Adds a key management parameter to the media.

10.1.47.3.6 - CSdpLevelMedia::AddOtherAttribute Method

Adds an unknown attribute in the media.

C++

```
void AddOtherAttribute(IN CSdpFieldAttributeOther& rOtherAttribute);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributeOther& rOtherAttribute`	unknown attribute to add.

Description

Adds a new unknown attribute at the end of the vector.

10.1.47.3.7 - CSdpLevelMedia::AddRtpmap Method

Adds a rtp map.

C++

```
void AddRtpmap(IN const CSdpFieldAttributeRtpmap& rRtpmap);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeRtpmap& rRtpmap`	Reference to a CSdpFieldAttributeRtpmap (see page 187).

Description

Adds a RtpMap to the list and calls Validate (see page 324).

10.1.47.3.8 - CSdpLevelMedia::ClearExplicitConnectionDatas Method

Clears all connection data of the media level.

C++

```
void ClearExplicitConnectionDatas();
```

Description

Clears all the explicit connection data.

10.1.47.3.9 - CSdpLevelMedia::FindCandidate Method

Searches for a candidate with the corresponding address and port.

C++

```
const CSdpFieldAttributeIceCandidate* FindCandidate(IN const char* pszAddress, IN unsigned int uPort) const;
```

Parameters

Parameters	Description
`IN const char* pszAddress`	Address of the candidate to search.
`IN unsigned int uPort`	Port of the candidate to search.

Returns

A pointer to the found CSdpFieldAttributeIceCandidate (see page 106) or NULL if no corresponding ICE candidate is found.

Description

Searches for an ICE candidate with the corresponding address and port.

See Also

CSdpFieldAttributeIceCandidate (see page 106)

10.1.47.3.10 - CSdpLevelMedia::FindRtpMapIndexByEncodingName Method

Finds the rtpmap index for the given encoding name.

C++

```
int FindRtpMapIndexByEncodingName(IN const char* pszEncodingName) const;
```

Parameters

Parameters	Description
`IN const char* pszEncodingName`	The rtp encoding name to find.

Returns

Rtp map index or -1 when not found.

Description

Finds the provided encoding name from the internal stream's rtp map vector.

10.1.47.3.11 - GetBandwidth

10.1.47.3.11.1 - CSdpLevelMedia::GetBandwidth Method

Gets the vector of Bandwidth field, i.e. "b=" field.

C++

```
CVector<CString>& GetBandwidth();
const CVector<CString>& GetBandwidth() const;
```

Returns

Reference to the vector of bandwidth fields.

Description

Gets the vector of bandwidth fields (i.e. "b=" fields).

10.1.47.3.12 - GetConfPreCondVector

10.1.47.3.12.1 - CSdpLevelMedia::GetConfPreCondVector Method

Gets the vector of CONF precondition. i.e. "a=conf" field.

C++

```
CVector<CSdpFieldAttributePreCondConf>& GetConfPreCondVector();
const CVector<CSdpFieldAttributePreCondConf>& GetConfPreCondVector() const;
```

Returns

The CONF precondition vector.

Description

Returns the CONF precondition vector.

10.1.47.3.13 - GetConnectionData

10.1.47.3.13.1 - CSdpLevelMedia::GetConnectionData Method

Gets the connection data at the specified index.

C++

```
CSdpFieldConnectionData& GetConnectionData(IN uint16_t uIndex);
const CSdpFieldConnectionData& GetConnectionData(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the connection data to get.

Returns

Reference to to the connection data field attribute.

Description

This method gets the CSdpFieldConnectionData (see page 228).

10.1.47.3.14 - GetCrypto

10.1.47.3.14.1 - CSdpLevelMedia::GetCrypto Method

Gets the Crypto stored at the specified index.

C++

```
CSdpFieldAttributeCrypto& GetCrypto(IN unsigned int uIndex);
const CSdpFieldAttributeCrypto& GetCrypto(IN unsigned int uIndex) const;
```

Parameters

Parameters	Description
`IN unsigned int uIndex`	Index of the element stored in the Crypto vector.

Returns

Reference to the requested CSdpFieldAttributeCrypto (see page 84).

Description

Provides one Crypto stored in the session.

10.1.47.3.15 - GetCurrPreCondVector

10.1.47.3.15.1 - CSdpLevelMedia::GetCurrPreCondVector Method

Gets the vector of CURR precondition. i.e. "a=curr" field.

C++

```
CVector<CSdpFieldAttributePreCondCurr>& GetCurrPreCondVector();
const CVector<CSdpFieldAttributePreCondCurr>& GetCurrPreCondVector() const;
```

Returns

The CURR precondition vector.

Description

Returns the CURR precondition vector.

10.1.47.3.16 - GetDesPreCondVector

10.1.47.3.16.1 - CSdpLevelMedia::GetDesPreCondVector Method

Gets the vector of DES precondition. i.e. "a=des" field.

C++

```
CVector<CSdpFieldAttributePreCondDes>& GetDesPreCondVector();
const CVector<CSdpFieldAttributePreCondDes>& GetDesPreCondVector() const;
```

Returns

The DES precondition vector.

Description

Returns the DES precondition vector.

10.1.47.3.17 - CSdpLevelMedia::GetDirection Method

Gets the direction stored in the media level or in the session.

C++

```
EAttributeType GetDirection() const;
```

Returns

The EAttributeType currently configured in the level media or in the session.

Description

Returns the direction stored in the media level or in the session.

10.1.47.3.18 - GetEncodingNameFromPayloadType

10.1.47.3.18.1 - CSdpLevelMedia::GetEncodingNameFromPayloadType Method

Searches the encoding name from the payload type in the rtpmap attributes.

C++

```
void GetEncodingNameFromPayloadType(IN const CString& rstrPayloadType, OUT CString& rstr) const;
```

Parameters

Parameters	Description
`IN const CString& rstrPayloadType`	The payload type for which to search in the rtpmap attributes.
`OUT CString& rstr`	OUT param. The string in which the corresponding encoding is put. The string is cleared. If the payload type was not found, the string is empty.

Description

This method returns the encoding name corresponding to the payload type in a rtpmap attribute.

It gets the encoding name from the rtpmap attributes. If the payload type is not found in any rtpmap, an empty string is returned.

Warning

Due to a limitation in the rtpmap class, this method only supports rstrPayloadType being a numeric value.

10.1.47.3.18.2 - CSdpLevelMedia::GetEncodingNameFromPayloadType Method

Searches the encoding name from the payload type in the rtpmap attributes.

C++

```
void GetEncodingNameFromPayloadType(IN uint32_t uPayloadType, OUT CString& rstr) const;
```

Parameters

Parameters	Description
`IN uint32_t uPayloadType`	The payload type for which to search in the rtpmap attributes.
`OUT CString& rstr`	The string in which the corresponding encoding is put. The string is cleared. If the payload type was not found, the string is empty.

Description

This method returns the encoding name corresponding to the payload type in a rtpmap attribute.

It gets the encoding name from the rtpmap attributes. If the payload type is not found in any rtpmap, an empty string is returned.

10.1.47.3.19 - CSdpLevelMedia::GetEncryptionKey Method

Gets the Encryption key field, i.e. "k=" field.

C++

```
const CString& GetEncryptionKey() const;
```

Returns

Constant reference to the encryption key string.

Description

Gets the encryption key string.

10.1.47.3.20 - GetExplicitConnectionData

10.1.47.3.20.1 - CSdpLevelMedia::GetExplicitConnectionData Method

Gets the connection data at the specified index.

C++

```
CSdpFieldConnectionData& GetExplicitConnectionData(IN uint16_t uIndex);
const CSdpFieldConnectionData& GetExplicitConnectionData(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the connection data to get in the media.

Returns

Reference to the connection data field.

Description

This method returns the CSdpFieldConnectionData (see page 228) at the provided uIndex.

10.1.47.3.21 - CSdpLevelMedia::GetExplicitDirection Method

Gets the direction stored in the media level.

C++

```
EAttributeType GetExplicitDirection() const;
```

Returns

The EAttributeType currently configured in the level media.

Description

Returns the direction stored in the media level.

10.1.47.3.22 - GetFillBitRemoval

10.1.47.3.22.1 - CSdpLevelMedia::GetFillBitRemoval Method

Gets the fillbit removal field attribute.

C++

```
CSdpFieldAttributeFillBitRemoval& GetFillBitRemoval();
```

Returns

Reference to the fill bit removal field attribute.

Description

Gets the fill bit removal field attribute.

10.1.47.3.23 - GetFmtp

10.1.47.3.23.1 - CSdpLevelMedia::GetFmtp Method

Gets the Fmtp attribute at the specified index.

C++

```
CSdpFieldAttributeFmtp& GetFmtp(IN uint16_t uIndex);
const CSdpFieldAttributeFmtp& GetFmtp(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the element stored in the Fmtps vector.

Returns

Reference to the requested CSdpFieldAttributeFmtp (see page 95).

Description

Gets the Fmtps attributes in the media at the provided index.

10.1.47.3.24 - GetFmtpFromEncoding

10.1.47.3.24.1 - CSdpLevelMedia::GetFmtpFromEncoding Method

Gets the Fmtp attribute from the compression algorithm.

C++

```
CSdpFieldAttributeFmtp* GetFmtpFromEncoding(IN ERtpCompressionAlgorithm eEncoding);
const CSdpFieldAttributeFmtp* GetFmtpFromEncoding(IN ERtpCompressionAlgorithm eEncoding) const;
```

Parameters

Parameters	Description
`IN ERtpCompressionAlgorithm eEncoding`	The compression algorithm used from ERtpCompressionAlgorithm enum.

Returns

The first fmtp attribute found for payload type corresponding to the encoding name.

NULL if the encoding name was not found in a rtpmap.

NULL if the encoding name was found in a rtpmap but a fmtp attribute was not found with the corresponding payload type.

Description

This method returns the first fmtp attribute with the same encoding name.

See Also

GetFmtpFromPayloadType (see page 300)

10.1.47.3.24.2 - CSdpLevelMedia::GetFmtpFromEncoding Method

Gets the Fmtp attribute from the compression algorithm name.

C++

```
CSdpFieldAttributeFmtp* GetFmtpFromEncoding(IN const char* pszEncodingName);
const CSdpFieldAttributeFmtp* GetFmtpFromEncoding(IN const char* pszEncodingName) const;
```

Parameters

Parameters	Description
szEncodingName	The encoding name used in the rtpmap to map the payload type.

Returns

The first fmtp attribute found for payload type corresponding to the encoding name.

NULL if the encoding name was not found in a rtpmap.

NULL if the encoding name was found in a rtpmap but a fmtp attribute was not found with the corresponding payload type.

Description

This method returns the first fmtp attribute with the same encoding name.

If the encoding name to be searched is static, use GetFmtpFromPayloadType (see page 300) instead since no rtpmap is necessary for these encoding names.

See Also

GetFmtpFromPayloadType (see page 300)

10.1.47.3.25 - GetFmtpFromPayloadType

10.1.47.3.25.1 - CSdpLevelMedia::GetFmtpFromPayloadType Method

Gets the Fmtp attribute from the payload type.

C++

```
CSdpFieldAttributeFmtp* GetFmtpFromPayloadType(IN uint32_t uPayloadType);
const CSdpFieldAttributeFmtp* GetFmtpFromPayloadType(IN uint32_t uPayloadType) const;
```

Parameters

Parameters	Description
IN uint32_t uPayloadType	The payload type contained in the fmtp attribute.

Returns

The first fmtp attribute found with the payload type.

NULL if no fmtp attribute was found with the payload type.

Description

This method returns the first fmtp attribute with the specified payload type.

If the encoding name to be searched is dynamic, use GetFmtpFromEncoding (see page 299) instead since there is no payload type officialy assigned to these encoding names.

See Also

GetFmtpFromEncoding (see page 299)

10.1.47.3.26 - GetFmtpRedundancy

10.1.47.3.26.1 - CSdpLevelMedia::GetFmtpRedundancy Method

Gets the Fmtp redundancy attribute.

C++

```
CSdpFmtpRedundancy* GetFmtpRedundancy();
const CSdpFmtpRedundancy* GetFmtpRedundancy() const;
```

Returns

Pointer to the internal GetFmtpRedundancy.

Description

Provides access to the Fmtp redundancy stored in the session.

See Also

GetFmtpFromEncoding (see page 299)

10.1.47.3.27 - GetFmtpTelEvent

10.1.47.3.27.1 - CSdpLevelMedia::GetFmtpTelEvent Method

Gets the Fmtp telephone event attribute.

C++

```
CSdpFmtpTelEvent* GetFmtpTelEvent();
const CSdpFmtpTelEvent* GetFmtpTelEvent() const;
```

Returns

Pointer to the internal CSdpFmtpTelEvent (see page 274).

Description

Provides access to the Fmtp Telephone Event stored in the session.

See Also

GetFmtpFromEncoding (see page 299)

10.1.47.3.28 - GetIceCandidate

10.1.47.3.28.1 - CSdpLevelMedia::GetIceCandidate Method

Returns a reference to the list of ICE candidate.

C++

```
CVector<CSdpFieldAttributeIceCandidate>& GetIceCandidate();
const CVector<CSdpFieldAttributeIceCandidate>& GetIceCandidate() const;
```

Returns

A reference to the vector of ICE candidate.

Description

Returns a reference to the vector of ICE candidate.

See Also

ValidateIceCandidates (see page 325)

10.1.47.3.29 - GetIcePassword

10.1.47.3.29.1 - CSdpLevelMedia::GetIcePassword Method

Returns the ice-pwd attribute.

C++

```
CSdpFieldAttributeIcePwd& GetIcePassword();
const CSdpFieldAttributeIcePwd& GetIcePassword() const;
```

Returns

A reference to CSdpFieldAttributeIcePwd (see page 118).

Description

Returns a reference to the ICE password.

10.1.47.3.30 - GetIceRemoteCandidatesAttribute

10.1.47.3.30.1 - CSdpLevelMedia::GetIceRemoteCandidatesAttribute Method

Returns a reference to the ICE remote-candidates attribute.

C++

```
CSdpFieldAttributeIceRemoteCandidates& GetIceRemoteCandidatesAttribute();
const CSdpFieldAttributeIceRemoteCandidates& GetIceRemoteCandidatesAttribute() const;
```

Returns

A reference to ICE remote-candidates attribute.

Description

Returns a reference to the ICE remote-candidates attribute.

10.1.47.3.31 - GetIceUserFragment

10.1.47.3.31.1 - CSdpLevelMedia::GetIceUserFragment Method

Returns the ice-ufrag attribute.

C++

```
CSdpFieldAttributeIceUserFrag& GetIceUserFragment();
const CSdpFieldAttributeIceUserFrag& GetIceUserFragment() const;
```

Returns

A reference to CSdpFieldAttributeIceUserFrag (see page 131).

Description

Returns a reference to the ICE user fragment.

10.1.47.3.32 - CSdpLevelMedia::GetInformation Method

Gets the information field, i.e. "i=" field.

C++

```
const CString& GetInformation() const;
```

Returns

Constant reference to the information string member.

Description

Gets the information string member.

10.1.47.3.33 - GetKeyMgmt

10.1.47.3.33.1 - CSdpLevelMedia::GetKeyMgmt Method

Gets the key management at the specified index.

C++

```
CSdpFieldAttributeKeyMgmt& GetKeyMgmt(IN uint16_t uIndex);
const CSdpFieldAttributeKeyMgmt& GetKeyMgmt(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the key management attribute to get.

Returns

The key management attribute.

Description

Gets the key management attribute at the specified index.

10.1.47.3.34 - GetKeyMgmtParam

10.1.47.3.34.1 - CSdpLevelMedia::GetKeyMgmtParam Method

Gets the key management parameter at the specified index.

C++

```
CSdpKeyManagementParameter& GetKeyMgmtParam(IN uint16_t uIndex);
const CSdpKeyManagementParameter& GetKeyMgmtParam(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the key management parameter to get.

Returns

The key management parameter.

Description

Gets the key management parameter for the specified index.

10.1.47.3.35 - GetMaxBitRate

10.1.47.3.35.1 - CSdpLevelMedia::GetMaxBitRate Method

Gets the maximum bitrate field attribute.

C++

```
CSdpFieldAttributeMaxBitRate& GetMaxBitRate();
const CSdpFieldAttributeMaxBitRate& GetMaxBitRate() const;
```

Returns

Reference to the max bitrate field attribute.

Description

Gets the max bitrate field attribute.

10.1.47.3.36 - GetMaxDatagram

10.1.47.3.36.1 - CSdpLevelMedia::GetMaxDatagram Method

Gets the maximum datagram field attribute.

C++

```
CSdpFieldAttributeMaxDatagram& GetMaxDatagram();
const CSdpFieldAttributeMaxDatagram& GetMaxDatagram() const;
```

Returns

Reference to the max datagram field attribute.

Description

Gets the max datagram field attribute.

10.1.47.3.37 - GetMediaAnnouncement

10.1.47.3.37.1 - CSdpLevelMedia::GetMediaAnnouncement Method

Provides access to the media announcement stored in the session.

C++

```
CSdpFieldMediaAnnouncement& GetMediaAnnouncement();
const CSdpFieldMediaAnnouncement& GetMediaAnnouncement() const;
```

Returns

Reference to the internal CSdpFieldMediaAnnouncement (see page 235).

Description

Provides access to the media announcement stored in the session.

See Also

SetMediaAnnouncement (see page 321)

10.1.47.3.38 - GetMid

10.1.47.3.38.1 - CSdpLevelMedia::GetMid Method

Gets the mid attribute field, i.e. "a=mid:" field.

C++

```
CSdpFieldAttributeMid& GetMid();
const CSdpFieldAttributeMid& GetMid() const;
```

Returns

The mid attribute field.

Description

Returns the mid attribute field.

10.1.47.3.39 - GetMptimeVector

10.1.47.3.39.1 - CSdpLevelMedia::GetMptimeVector Method

Gets the vector of mptime.

C++

```
CVector<unsigned int>& GetMptimeVector();
```

Returns

Reference to the vector of mptime.

Description

Gets the vector of the elements of the attribute mptime.

10.1.47.3.39.2 - CSdpLevelMedia::GetMptimeVector Method

Gets the vector of mptime.

C++

```
const CVector<unsigned int>& GetMptimeVector() const;
```

Returns

Reference to the vector of mptime.

Description

Gets the vector of the elements of the attribute mptime.

10.1.47.3.40 - CSdpLevelMedia::GetNbConnectionDatas Method

Gets the number of connection data. Default value if no data.

C++

```
uint32_t GetNbConnectionDatas() const;
```

Returns

The number of connection data.

Description

Provides the number of connection data. In the case the media does not have any "c=" field, the default value in the session is used.

See Also

GetNbExplicitConnectionDatas (see page 305)

10.1.47.3.41 - CSdpLevelMedia::GetNbCrypto Method

Gets the number of Crypto stored in the session.

C++

```
uint32_t GetNbCrypto() const;
```

Returns

The number of CSdpFieldAttributeCrypto (see page 84) stored.

Description

Provides the number of Crypto stored in the session.

See Also

GetNbConnectionDatas (see page 305)

10.1.47.3.42 - CSdpLevelMedia::GetNbExplicitConnectionDatas Method

Provides the exact number of connection data.

C++

```
uint32_t GetNbExplicitConnectionDatas() const;
```

Returns

The number of connection data.

Description

Provides the number of connection data. In the case the media does not have any "c=" field, the value 0 is returned.

See Also

GetNbConnectionDatas (see page 305)

10.1.47.3.43 - CSdpLevelMedia::GetNbFmtps Method

Gets the number of Fmtp in the media.

C++

```
uint32_t GetNbFmtps() const;
```

Returns

The number of Fmtps attributes in the media.

Description

Gets the number of Fmtps attributes in the media.

10.1.47.3.44 - CSdpLevelMedia::GetNbKeyMgmt Method

Gets the number of key management attributes.

C++

```
uint32_t GetNbKeyMgmt() const;
```

Returns

The number of key management attributes in the media.

Description

Gets the number of key management attributes in the media.

10.1.47.3.45 - CSdpLevelMedia::GetNbKeyMgmtParam Method

Gets the number of key management parameters.

C++

```
uint32_t GetNbKeyMgmtParam() const;
```

Returns

The number of key management parameters in the media.

Description

Gets the number of key management parameters in the media.

10.1.47.3.46 - CSdpLevelMedia::GetNbOtherAttributes Method

Gets the number of unknown field attributes.

C++

```
uint32_t GetNbOtherAttributes() const;
```

Returns

Number of other field attributes.

Description

Gets the number of other field attributes.

10.1.47.3.47 - CSdpLevelMedia::GetNbParsedPtimes Method

Returns the number of parsed ptimes.

C++

```
unsigned int GetNbParsedPtimes() const;
```

Returns

Number of ptimes encountered while parsing.

Description

Provides access to the number of parsed ptimes. This is only an indication of how many ptimes were seen in the parsing process. Only the last ptime found is kept and returned by GetPtime (see page 308)().

See Also

GetPtime (see page 308)

10.1.47.3.48 - CSdpLevelMedia::GetNbRtpmaps Method

Gets the number of rtp maps.

C++

```
uint32_t GetNbRtpmaps() const;
```

Returns

The number of Rtp maps.

Description

Provides the number of rtp maps stored in the session.

See Also

GetNbConnectionDatas (see page 305)

10.1.47.3.49 - GetOtherAttribute

10.1.47.3.49.1 - CSdpLevelMedia::GetOtherAttribute Method

Gets the unknown field attribute at the specified index.

C++

```
CSdpFieldAttributeOther& GetOtherAttribute(IN uint16_t uIndex);
const CSdpFieldAttributeOther& GetOtherAttribute(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the unknown attribute to get.

Returns

Reference to the requested unknown attribute.

Description

Gets the unknown attribute at the specified index.

10.1.47.3.50 - CSdpLevelMedia::GetOtherAttributes Method

Gets the unknown field attributes.

C++

```
CVector<CSdpFieldAttributeOther>& GetOtherAttributes();
```

Returns

Reference to the unknown attribute vector.

Description

Gets the unknown attribute vector.

10.1.47.3.51 - GetPayloadTypeFromEncoding

10.1.47.3.51.1 - CSdpLevelMedia::GetPayloadTypeFromEncoding Method

Searches the payload type from the encoding name in the rtpmap attributes.

C++

```
uint32_t GetPayloadTypeFromEncoding(IN const char* pszEncodingName) const;
```

Parameters

Parameters	Description
szEncodingName	The encoding name used in the rtpmap to map the payload type.

Returns

The payload type that corresponds to the encoding name.

CSdpFieldAttributeFmtp::uINVALID_MEDIA_FORMAT (see page 96); if the encoding name was not found in a rtpmap.

Description

This method returns the payload type corresponding to the encoding name in a rtpmap attribute.

It gets the payload type from the rtpmap attributes. If the encoding name is not found in any rtpmap, CSdpFieldAttributeFmtp::uINVALID_MEDIA_FORMAT (see page 96) is returned.

10.1.47.3.51.2 - CSdpLevelMedia::GetPayloadTypeFromEncoding Method

Searches the payload type from the encoding name in the rtpmap attributes.

C++

```
void GetPayloadTypeFromEncoding(IN const char* pszEncodingName, OUT CString& rstrPayloadType) const;
```

Parameters

Parameters	Description
OUT CString& rstrPayloadType	OUT value. Left empty if the method fails.
szEncodingName	The encoding name used in the rtpmap to map the payload type.

Description

This method returns the payload type corresponding to the encoding name in a rtpmap attribute.

It gets the payload type from the rtpmap attributes. If the encoding name is not found in any rtpmap, rstrPayloadType is left empty.

Warning

Due to a limitation in the rtpmap class, this method only supports pszEncodingName being a numeric value.

10.1.47.3.52 - GetPtime

10.1.47.3.52.1 - CSdpLevelMedia::GetPtime Method

Gets the last seen ptimes in the parsing process.

C++

```
CSdpFieldAttributePtime& GetPtime();
const CSdpFieldAttributePtime& GetPtime() const;
const CSdpFieldAttributeFillBitRemoval& GetFillBitRemoval() const;
```

Returns

Last ptime found seen in the parsing process.

Description

Provides access to the last seen ptimes in the parsing process.

See Also

GetNbParsedPtimes (see page 306)

10.1.47.3.53 - GetRtpmap

10.1.47.3.53.1 - CSdpLevelMedia::GetRtpmap Method

Gets the rtp map at the specified index.

C++

```
CSdpFieldAttributeRtpmap& GetRtpmap(IN uint16_t uIndex);
const CSdpFieldAttributeRtpmap& GetRtpmap(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
IN uint16_t uIndex	Index of the element stored in the RtpMaps vector.

Returns

Reference to the requested CSdpFieldAttributeRtpmap (see page 187).

Description

Provides one RtpMap stored in the session.

10.1.47.3.54 - CSdpLevelMedia::GetSdpFieldAttributeRtcp Method

Gets the Rtcp field attribute.

C++

```
const CSdpFieldAttributeRtcp& GetSdpFieldAttributeRtcp() const;
```

Returns

Reference to the rtcp attribute.

Description

Gets the unknown rtcp attribute (i.e. the "a=rtcp:" field).

10.1.47.3.55 - GetSession

10.1.47.3.55.1 - CSdpLevelMedia::GetSession Method

Gets the session lvel.

C++

```
CSdpLevelSession* GetSession();
const CSdpLevelSession* GetSession() const;
```

Returns

Pointer to a CSdpLevelSession (see page 326).

Description

Gets the level session associated with this level media.

10.1.47.3.56 - GetSilenceSuppressionSupport

10.1.47.3.56.1 - CSdpLevelMedia::GetSilenceSuppressionSupport Method

Gets the silence suppression attribute from this media.

C++

```
const CSdpFieldAttributeSilenceSupp& GetSilenceSuppressionSupport() const;
CSdpFieldAttributeSilenceSupp& GetSilenceSuppressionSupport();
```

Returns

A const reference to the silence suppression attribute.

Description

Returns the silence suppression attribute found in the media. This attribute may be empty or invalid. It is up to the application to validate the contents of the attribute.

Notes: Use GetSilenceSupp returning a non const reference to set the value of the CSdpFieldAttributeSilenceSupp (see page 194) in the CSdpLevelMedia (see page 288).

10.1.47.3.57 - GetT38ErrorControl

10.1.47.3.57.1 - CSdpLevelMedia::GetT38ErrorControl Method

Gets the T38 error control field attribute.

C++

```
CSdpFieldAttributeT38ErrorControl& GetT38ErrorControl();
const CSdpFieldAttributeT38ErrorControl& GetT38ErrorControl() const;
```

Returns

Reference to the T38 error control field attribute.

Description

Gets the T38 error control field attribute.

10.1.47.3.58 - GetT38FacsimileMaxBuffer

10.1.47.3.58.1 - CSdpLevelMedia::GetT38FacsimileMaxBuffer Method

Gets the T38 Facsimile maximum buffer field attribute.

C++

```
CSdpFieldAttributeT38FacsimileMaxBuffer& GetT38FacsimileMaxBuffer();
const CSdpFieldAttributeT38FacsimileMaxBuffer& GetT38FacsimileMaxBuffer() const;
```

Returns

Reference to the T38 facsimile max buffer field attribute.

Description

Gets the T38 facsimile max buffer field attribute.

10.1.47.3.59 - GetT38FacsimileRateMgmnt

10.1.47.3.59.1 - CSdpLevelMedia::GetT38FacsimileRateMgmnt Method

Gets the T38 Facsimile rate management field attribute.

C++

```
CSdpFieldAttributeT38FacsimileRateMgmnt& GetT38FacsimileRateMgmnt();
const CSdpFieldAttributeT38FacsimileRateMgmnt& GetT38FacsimileRateMgmnt() const;
```

Returns

Reference to the T38 facsimile rate management field attribute.

Description

Gets the T38 facsimile rate management field attribute.

10.1.47.3.60 - GetTranscodingJBIG

10.1.47.3.60.1 - CSdpLevelMedia::GetTranscodingJBIG Method

Gets the transcoding JBIG field attribute.

C++

```
CSdpFieldAttributeTranscodingJBIG& GetTranscodingJBIG();
const CSdpFieldAttributeTranscodingJBIG& GetTranscodingJBIG() const;
```

Returns

Reference to the transcoding JBIG field attribute.

Description

Gets the transcoding JBIG field attribute.

10.1.47.3.61 - GetTranscodingMMR

10.1.47.3.61.1 - CSdpLevelMedia::GetTranscodingMMR Method

Gets the transcoding MMR field attribute.

C++

```
CSdpFieldAttributeTranscodingMMR& GetTranscodingMMR();
const CSdpFieldAttributeTranscodingMMR& GetTranscodingMMR() const;
```

Returns

Reference to the transcoding MMR field attribute.

Description

Gets the transcoding MMR field attribute.

10.1.47.3.62 - GetVersion

10.1.47.3.62.1 - CSdpLevelMedia::GetVersion Method

Gets the version field attribute.

C++

```
CSdpFieldAttributeVersion& GetVersion();
const CSdpFieldAttributeVersion& GetVersion() const;
```

Returns

Reference to the version field attribute.

Description

Gets the version field attribute.

10.1.47.3.63 - CSdpLevelMedia::InsertCrypto Method

Inserts the crypto attribute in the media at the specified index.

C++

```
void InsertCrypto(IN unsigned int uIndex, IN const CSdpFieldAttributeCrypto& rCrypto);
```

Parameters

Parameters	Description
`IN unsigned int uIndex`	Position in the vector where to insert the new Crypto.
`IN const CSdpFieldAttributeCrypto& rCrypto`	Reference to a CSdpFieldAttributeCrypto (see page 84) to store in the session.

Description

Inserts the provided Crypto in the session at the provided index in the vector.

10.1.47.3.64 - CSdpLevelMedia::InsertRtpmap Method

Inserts a rtp map at the specified index.

C++

```
void InsertRtpmap(IN uint16_t uIndex, IN const CSdpFieldAttributeRtpmap& rRtpmap);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Position index in the RtpMaps vector.
`IN const CSdpFieldAttributeRtpmap& rRtpmap`	Reference to a CSdpFieldAttributeRtpmap (see page 187).

Description

Inserts the given CSdpFieldAttributeRtpmap (see page 187) at the given index uIndex in the RtpMaps vector.

10.1.47.3.65 - CSdpLevelMedia::IsExplicitInactive Method

Verifies whether or not the direction is set to inactive or if the session is inactive.

C++

```
bool IsExplicitInactive() const;
```

Returns

True if the direction is inactive or if the direction is unknown and the session is inactive. False otherwise.

Description

Verifies whether or not the direction is set to inactive or if the session is inactive

10.1.47.3.66 - CSdpLevelMedia::IsExplicitRecvOnly Method

Verifies whether or not the direction is set to receive only.

C++

```
bool IsExplicitRecvOnly() const;
```

Returns

True if the direction is receive only. False otherwise.

Description

Verifies whether or not the direction is set to receive only. Does not care about the session receive only flag.

10.1.47.3.67 - CSdpLevelMedia::IsExplicitSendOnly Method

Verifies whether or not the direction is set to send only.

C++

```
bool IsExplicitSendOnly() const;
```

Returns

True if the direction is send only. False otherwise.

Description

Verifies whether or not the direction is set to send only. Does not care about the session send only flag.

10.1.47.3.68 - CSdpLevelMedia::IsExplicitSendRecv Method

Verifies whether or not the direction is set to send and receive.

C++

```
bool IsExplicitSendRecv() const;
```

Returns

True if the direction is send and receive. False otherwise.

Description

Verifies whether or not the direction is set to send and receive. Does not care about the session send and receive flag.

10.1.47.3.69 - CSdpLevelMedia::IsIceAttributePresent Method

Returns true if the media contains at least one ICE attribute.

C++

```
bool IsIceAttributePresent() const;
```

Returns

true if the media contains at least one ICE related attributes.

Description

Returns true if the media contains at least one of the ICE attributes: ice-ufrag, ice-pwd, candidate, remote-candidates or ice-mismatch.

10.1.47.3.70 - CSdpLevelMedia::IsIceMismatch Method

The attribute a=ice-mismatch is present.

C++

```
bool IsIceMismatch() const;
```

Returns

- true: If a a=ice-mismatch is present.
- false: Otherwise.

Description

Returns true if a a=ice-mismatch is present in the media.

10.1.47.3.71 - CSdpLevelMedia::IsInactive Method

Verifies whether or not the direction is set to inactive.

C++

```
bool IsInactive() const;
```

Returns

True if the direction is inactive. False otherwise.

Description

Verifies whether or not the direction is set to inactive. Does not care about the session send and receive flag.

10.1.47.3.72 - CSdpLevelMedia::IsMicroLitePortPresent Method

Checks that all candidates have the microliteport extension.

C++

```
bool IsMicroLitePortPresent() const;
```

Returns

- true: present in all candidates.
- false otherwise.

Description

Checks that there is at least one candidate and that all candidates have the microliteport extension.

10.1.47.3.73 - CSdpLevelMedia::IsRecvOnly Method

Verifies whether or not the direction or the session is set to receive only.

C++

```
bool IsRecvOnly() const;
```

Returns

True if the direction is receive only or if the direction is unknown and the session is receive only. False otherwise.

Description

Verifies whether or not the direction is set to receive only or if the session is receive only.

10.1.47.3.74 - CSdpLevelMedia::IsRtcpDeactivated Method

Checks if RTCP is disabled for this media.

C++

```
bool IsRtcpDeactivated(OUT bool* pbRrPresent = NULL, OUT bool* pbRsPresent = NULL) const;
```

Parameters

Parameters	Description
`OUT bool* pbRrPresent = NULL`	Pointer to a bool where the result is stored. Indicates if b=RR:0 is present.
`OUT bool* pbRsPresent = NULL`	Pointer to a bool where the result is stored. Indicates if b=RS:0 is present.

Returns

true if the media has RTCP deactivated.

Description

Returns true if the media or session contains both bandwidth modifiers b=RS:0 and b=RR:0 (RFC3556). The bandwidth modifiers can be present at both media or session level. The method thus also searches in bandwidth at session level. This indicates that RTCP is deactivated for this media.

10.1.47.3.75 - CSdpLevelMedia::IsRtcpMuxPresent Method

Returns if a rtcp-mux attribute is present.

C++

```
bool IsRtcpMuxPresent() const;
```

Returns

- true if rtcp-mux attribute is present.
- false if it is not present.

Description

Returns if the rtcp-mux attribute is present or not.

10.1.47.3.76 - CSdpLevelMedia::IsSendOnly Method

Verifies whether or not the direction or the session is set to send only.

C++

```
bool IsSendOnly() const;
```

Returns

True if the direction is send only or if the direction is unknown and the session is send only. False otherwise.

Description

Verifies whether or not the direction is set to send only or if the session is send only.

10.1.47.3.77 - CSdpLevelMedia::IsSendRecv Method

Verifies whether or not the direction or the session is set to send and receive.

C++

```
bool IsSendRecv() const;
```

Returns

True if the direction is send and receive or if the direction is unknown and the session is send and receive. False otherwise.

Description

Verifies whether or not the direction is set to send and receive or if the session is send and receive

10.1.47.3.78 - CSdpLevelMedia::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this media.

10.1.47.3.79 - CSdpLevelMedia::ParseKeyMgmt Method

Parses all the needed information for the key management.

C++

```
EParserResult ParseKeyMgmt(IN const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`IN const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing.

Description

Parses the key management attribute. The attribute parsed will create an object of the appropriate type if necessary.

10.1.47.3.80 - CSdpLevelMedia::RemoveCrypto Method

Removes the crypto attribute in the media at the specified index.

C++

```
void RemoveCrypto(IN unsigned int uIndex);
```

Parameters

Parameters	Description
`IN unsigned int uIndex`	Position in the vector where to delete the new Crypto.

Description

Erases the Crypto stored at the provided index.

10.1.47.3.81 - CSdpLevelMedia::RemoveFmtp Method

Removes a fmtp at the specified index.

C++

```
void RemoveFmtp(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Position index in the Fmtps vector.

Description

Erases the element located at index uIndex from the Fmtps vector.

10.1.47.3.82 - RemoveFmtpFromEncoding

10.1.47.3.82.1 - CSdpLevelMedia::RemoveFmtpFromEncoding Method

Removes fmtp from encoding from the specified compression algorithm.

C++

```
bool RemoveFmtpFromEncoding(IN ERtpCompressionAlgorithm eEncoding);
```

Parameters

Parameters	Description
`IN ERtpCompressionAlgorithm eEncoding`	The encoding used in the rtpmap to map the payload type.

Returns

true if a fmtp attribute for the encoding name was removed from the list.

false if the encoding name was found in a rtpmap but a fmtp attribute was not found with the corresponding payload type.

Description

This method removes the first fmtp attribute using the same encoding name.

If the encoding name to be searched is static, use RemoveFmtpFromPayloadType (see page 317) instead since no rtpmap is necessary for these encoding names.

See Also

RemoveFmtpFromPayloadType (see page 317)

10.1.47.3.82.2 - CSdpLevelMedia::RemoveFmtpFromEncoding Method

Removes fmtp from encoding from the specified encoding name.

C++

```
bool RemoveFmtpFromEncoding(IN const char* pszEncodingName);
```

Parameters

Parameters	Description
`IN const char* pszEncodingName`	The encoding name used in the rtpmap to map the payload type.

Returns

true if a fmtp attribute for the encoding name was removed from the list.

false if the encoding name was not found in a rtpmap.

false if the encoding name was found in a rtpmap but a fmtp attribute was not found with the corresponding payload type.

Description

This method removes the first fmtp attribute using the same encoding name.

If the encoding name to be searched is static, use RemoveFmtpFromPayloadType (see page 317) instead since no rtpmap is necessary for these encoding names.

See Also

RemoveFmtpFromPayloadType (see page 317)

10.1.47.3.83 - CSdpLevelMedia::RemoveFmtpFromPayloadType Method

Removes fmtp from the specified payload type.

C++

```
bool RemoveFmtpFromPayloadType(IN uint32_t uPayloadType);
```

Parameters

Parameters	Description
`IN uint32_t uPayloadType`	The payload type contained in the fmtp attribute to remove.

Returns

true if a fmtp attribute with the payload type was found and removed.

false if no fmtp attribute was found with the payload type.

Description

This method removes the first fmtp attribute with the specified payload type.

If the encoding name to be searched is dynamic, use RemoveFmtpFromEncoding (see page 316) instead since there is no payload type officialy assigned to these encoding names.

See Also

RemoveFmtpFromEncoding (see page 316)

10.1.47.3.84 - CSdpLevelMedia::RemoveFmtpRedundancy Method

Removes fmtp redundancy from the media.

C++

```
bool RemoveFmtpRedundancy();
```

Returns

true if a fmtp attribute eRED was removed from the list.

false if the encoding name was not found in a rtpmap.

Description

This method removes FMTP Redundancy.

See Also

RemoveFmtpFromPayloadType (see page 317), RemoveFmtpRedundancy, RemoveFmtpTelEvent (see page 317)

10.1.47.3.85 - CSdpLevelMedia::RemoveFmtpTelEvent Method

Removes fmtp telephone event from the media.

C++

```
bool RemoveFmtpTelEvent();
```

Returns

true if a fmtp attribute eTELEPHONE_EVENT was removed from the list.

false if the encoding name was not found in a rtpmap.

Description

This method removes telephone events.

See Also

RemoveFmtpFromPayloadType (see page 317), RemoveFmtpRedundancy (see page 317), RemoveFmtpTelEvent

10.1.47.3.86 - CSdpLevelMedia::RemoveKeyMgmt Method

Removes the key management at the specified index.

C++

```
void RemoveKeyMgmt(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the key management to remove.

Description

Removes the key management attribute at the specified index.

10.1.47.3.87 - CSdpLevelMedia::RemoveKeyMgmtParam Method

Removes the key management parameter at the specified index.

C++

```
void RemoveKeyMgmtParam(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the key management parameter to remove.

Description

Removes the key management parameter at the specified index.

10.1.47.3.88 - CSdpLevelMedia::RemoveRtpmap Method

Removes a rtp map at the specified index.

C++

```
void RemoveRtpmap(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Position index in the RtpMaps vector.

Description

Erases the element located at index uIndex from the RtpMaps vector then calls Validate (see page 324).

10.1.47.3.89 - CSdpLevelMedia::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 315).

10.1.47.3.90 - CSdpLevelMedia::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.47.3.91 - CSdpLevelMedia::SetDirection Method

Sets the direction flag.

C++

```
void SetDirection(IN EAttributeType eDirection);
```

Parameters

Parameters	Description
`IN EAttributeType eDirection`	Indicates the direction to be set.

Description

Sets the direction property according to the EAttributeType input.

10.1.47.3.92 - CSdpLevelMedia::SetEncryptionKey Method

Sets the Encryption key field, i.e. "k=" field.

C++

```
void SetEncryptionKey(IN const char* szEncryptionKey);
```

Parameters

Parameters	Description
`IN const char* szEncryptionKey`	The encryption key string to set.

Description

Sets the encryption key string.

10.1.47.3.93 - CSdpLevelMedia::SetFillBitRemoval Method

Sets the fill bit removal field attribute.

C++

```
void SetFillBitRemoval(IN CSdpFieldAttributeFillBitRemoval& rFillBitRemoval);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributeFillBitRemoval& rFillBitRemoval`	Reference to the fill bit removal attribute.

Description

Sets the internal CSdpFieldAttributeFillBitRemoval (see page 90).

10.1.47.3.94 - CSdpLevelMedia::SetIceMismatch Method

Sets the a=ice-mismatch attribute.

C++

```
void SetIceMismatch(IN bool bIceMismatch);
```

Parameters

Parameters	Description
`IN bool bIceMismatch`	Whether or not the a=ice-mismatch is present in the media level.

Description

Sets whether or not the a=ice-mismatch is present in the media level.

10.1.47.3.95 - CSdpLevelMedia::SetInactive Method

Sets the inactive flag.

C++

```
void SetInactive(IN bool bInactive);
```

Parameters

Parameters	Description
`IN bool bInactive`	Indicates that the direction is currently inactive.

Description

Sets the direction property to inactive.

10.1.47.3.96 - CSdpLevelMedia::SetInformation Method

Sets the information field, i.e. "i=" field.

C++

```
void SetInformation(IN const char* szInformation);
```

Parameters

Parameters	Description
`IN const char* szInformation`	Information to set.

Description

Sets the information string member.

10.1.47.3.97 - CSdpLevelMedia::SetMaxBitRate Method

Sets the maximum bit rate field attribute.

C++

```
void SetMaxBitRate(CSdpFieldAttributeMaxBitRate& rMaxBitRate);
```

Parameters

Parameters	Description
`CSdpFieldAttributeMaxBitRate& rMaxBitRate`	Reference to the maximum bit rate attribute.

Description

Sets the internal CSdpFieldAttributeMaxBitRate (see page 149).

10.1.47.3.98 - CSdpLevelMedia::SetMaxDatagram Method

Sets the maximum datagram field attribute.

C++

```
void SetMaxDatagram(CSdpFieldAttributeMaxDatagram& rMaxDatagram);
```

Parameters

Parameters	Description
`rMaxBitRate`	Reference to the maximum datagram attribute.

Description

Sets the internal CSdpFieldAttributeMaxDatagram (see page 153).

10.1.47.3.99 - CSdpLevelMedia::SetMediaAnnouncement Method

Sets the media announcement field.

C++

```
void SetMediaAnnouncement(IN CSdpFieldMediaAnnouncement& rMediaAnnouncement);
```

Parameters

Parameters	Description
`IN CSdpFieldMediaAnnouncement& rMediaAnnouncement`	Reference to a CSdpFieldMediaAnnouncement (see page 235).

Description

Sets the MediaAnnouncement.

10.1.47.3.100 - CSdpLevelMedia::SetMicroLiteDefaultFamily Method

Specify which address family must be prioritized for the default destination when sending an offer.

C++

```
void SetMicroLiteDefaultFamily(IN CSocketAddr::EAddressFamily eFamily);
```

10.1.47.3.101 - CSdpLevelMedia::SetPtime Method

Sets the ptime.

C++

```
void SetPtime(IN CSdpFieldAttributePtime& rPacketTime);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributePtime& rPacketTime`	Reference to the packet time attribute.

Description

Sets the internal CSdpFieldAttributePtime (see page 177).

10.1.47.3.102 - CSdpLevelMedia::SetRecvOnly Method

Sets the receive only flag.

C++

```
void SetRecvOnly(IN bool bRecvOnly);
```

Parameters

Parameters	Description
`IN bool bRecvOnly`	Indicates if the direction is receive only (true) or not (false).

Description

Sets the direction property to receive only.

10.1.47.3.103 - CSdpLevelMedia::SetRtcpMux Method

Returns if a rtcp-mux attribute is present.

C++

```
void SetRtcpMux(IN bool bRtcpMux);
```

Parameters

Parameters	Description
`IN bool bRtcpMux`	true if the rtcp-mux attribute is present.

Description

Sets if a rtcp-mux attribute is present in the media or not.

10.1.47.3.104 - CSdpLevelMedia::SetSdpFieldAttributeRtcp Method

Sets the rtcp field attribute.

C++

```
void SetSdpFieldAttributeRtcp(IN CSdpFieldAttributeRtcp& rRtcpAttribute);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributeRtcp& rRtcpAttribute`	Sdp field attribute to set.

Description

Sets a new Sdp field attribute.

10.1.47.3.105 - CSdpLevelMedia::SetSendDirection Method

Sets the sending direction.

C++

```
void SetSendDirection(IN bool bSend);
```

Parameters

Parameters	Description
`IN bool bSend`	Indicates the send direction.

Description

Sets the send direction flags.

10.1.47.3.106 - CSdpLevelMedia::SetSendOnly Method

Sets the send only flag.

C++

```
void SetSendOnly(IN bool bSendOnly);
```

Parameters

Parameters	Description
`IN bool bSendOnly`	Indicates if the direction is send only (true) or not (false).

Description

Sets the direction property to send only.

10.1.47.3.107 - CSdpLevelMedia::SetSendRecv Method

Sets the send and receive flag.

C++

```
void SetSendRecv(IN bool bSendRecv);
```

Parameters

Parameters	Description
`IN bool bSendRecv`	Indicates that the direction is both send and receive (true) or not (false).

Description

Sets the direction property to send and receive.

10.1.47.3.108 - CSdpLevelMedia::SetSession Method

Sets the session level.

C++

```
void SetSession(IN CSdpLevelSession* pSession);
```

Parameters

Parameters	Description
`IN CSdpLevelSession* pSession`	Pointer to the CSdpLevelSession (see page 326) to set.

Description

Sets the associated level session to the level media.

10.1.47.3.109 - CSdpLevelMedia::SetT38BooleanEncoding Method

Sets the T.38 boolean encoding method for the T38FaxFillBitRemoval, T38FaxTranscodingMMR, and T38FaxTranscodingJBIG attributes.

C++

```
void SetT38BooleanEncoding(bool bT38ImplicitEncoding);
```

Parameters

Parameters	Description
`bool bT38ImplicitEncoding`	Indicates if the encoding method is the implicit method (true) or the explicit method (false).

Description

Sets the T.38 boolean encoding method for the T38FaxFillBitRemoval, T38FaxTranscodingMMR, and T38FaxTranscodingJBIG attributes.

10.1.47.3.110 - CSdpLevelMedia::SetT38ErrorControl Method

Sets the T38 error control field attribute.

C++

```
void SetT38ErrorControl(CSdpFieldAttributeT38ErrorControl& rT38ErrorControl);
```

Parameters

Parameters	Description
`CSdpFieldAttributeT38ErrorControl& rT38ErrorControl`	Reference to the T38 error control attribute.

Description

Sets the internal CSdpFieldAttributeT38ErrorControl (see page 198).

10.1.47.3.111 - CSdpLevelMedia::SetT38FacsimileMaxBuffer Method

Sets the T38 Facsimile maximum buffer field attribute.

C++

```
void SetT38FacsimileMaxBuffer(CSdpFieldAttributeT38FacsimileMaxBuffer& rMaxBuffer);
```

Parameters

Parameters	Description
`CSdpFieldAttributeT38FacsimileMaxBuffer& rMaxBuffer`	Reference to the T38 facsimile maximum buffer attribute.

Description

Sets the internal CSdpFieldAttributeT38FacsimileMaxBuffer (see page 202).

10.1.47.3.112 - CSdpLevelMedia::SetT38FacsimileRateMgmnt Method

Sets the T38 Facsimile rate management field attribute.

C++

```
void SetT38FacsimileRateMgmnt(CSdpFieldAttributeT38FacsimileRateMgmnt& rRateMgmnt);
```

Parameters

Parameters	Description
`CSdpFieldAttributeT38FacsimileRateMgmnt& rRateMgmnt`	Reference to the T38 facsimile rate management attribute.

Description

Sets the internal CSdpFieldAttributeT38FacsimileRateMgmnt (see page 206).

10.1.47.3.113 - CSdpLevelMedia::SetTranscodingJBIG Method

Sets the transcoding JBIG field attribute.

C++

```
void SetTranscodingJBIG(CSdpFieldAttributeTranscodingJBIG& rTranscodingJBIG);
```

Parameters

Parameters	Description
`CSdpFieldAttributeTranscodingJBIG& rTranscodingJBIG`	Reference to the transcoding JBIG attribute.

Description

Sets the internal CSdpFieldAttributeTranscodingJBIG (see page 213).

10.1.47.3.114 - CSdpLevelMedia::SetTranscodingMMR Method

Sets the transcoding MMR field attribute.

C++

```
void SetTranscodingMMR(CSdpFieldAttributeTranscodingMMR& rTranscodingMMR);
```

Parameters

Parameters	Description
`CSdpFieldAttributeTranscodingMMR& rTranscodingMMR`	Reference to the transcoding MMR attribute.

Description

Sets the internal CSdpFieldAttributeTranscodingMMR (see page 218).

10.1.47.3.115 - CSdpLevelMedia::SetVersion Method

Sets the version field attribute.

C++

```
void SetVersion(CSdpFieldAttributeVersion& rVersion);
```

Parameters

Parameters	Description
`CSdpFieldAttributeVersion& rVersion`	Reference to the version attribute.

Description

Sets the internal CSdpFieldAttributeVersion (see page 224).

10.1.47.3.116 - CSdpLevelMedia::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.47.3.117 - CSdpLevelMedia::ValidateIceCandidates Method

Validates that the IP address used by the media is also an ICE candidate.

C++

```
bool ValidateIceCandidates() const;
```

Returns

true if all addresses have ICE candidates. false otherwise. true is returned if there is no ICE attributes at all or if the media port is 0.

Description

Validates that the connection addresses associated with the media port have an ICE candidate specified. If enabled, RTCP address and port are also validated. If no ICE attribute is present in the SDP offer, true is always returned. If the media port is 0, true is also returned. This method should be called before using candidates. CSdpCapabilitiesMgr::GenerateAnswer (see page 41) automatically adds an ice-mismatch attribute if this fails for a media.

See Also

CSdpCapabilitiesMgr::GenerateAnswer (see page 41)

10.1.47.4 - Operators

10.1.47.4.1 - CSdpLevelMedia::= Operator

Assignment operator.

C++

```
CSdpLevelMedia& operator =(IN const CSdpLevelMedia& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.47.4.2 - CSdpLevelMedia::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpLevelMedia& rFrom) const;
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rFrom`	A reference to the CSdpLevelMedia (see page 288) with which to compare.

Returns

True if both CSdpLevelMedia (see page 288) are equal, false otherwise.

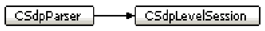
Description

Compares the two CSdpLevelMedia (see page 288) to see if they are equal.

10.1.48 - CSdpLevelSession Class

This class implements an abstraction of the level-session.

Class Hierarchy



C++

```
class CSdpLevelSession : public CSdpParser;
```

Description

This class is an abstraction of the level-session part of a SDP packet. It is used to Set and Get several fields defined in the BNF of RFC 2327.

These fields are: Protocol Version, Origin, Time, Attributes and Media Announcements. There can be several types of attributes.

RFC 2327 BNF:

```
level-session =  proto-version
                 origin-field
                 session-name-field
                 key-management-field
                 information-field
                 uri-field
                 email-fields
                 phone-fields
                 connection-field
                 bandwidth-fields
                 time-fields
                 key-field
                 attribute-fields
                 media-descriptions
```

The parser also supports group field attribute implemented by CSdpFieldAttributeGroup (see page 101). It is defined in RFC 3388.

Fields with no links are not implemented yet, they are ignored when they are parsed.

Location

SdpParser/CSdpLevelSession.h

See Also

CSdpFieldProtocolVersion (see page 255), CSdpFieldOrigin (see page 243), CSdpFieldSessionName (see page 259), CSdpFieldConnectionData (see page 228), CSdpFieldAttributeOther (see page 160), CSdpLevelMedia (see page 288), CSdpFieldAttributeGroup (see page 101)

Constructors

Constructor	Description
◆ CSdpLevelSession (see page 328)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpLevelSession (see page 329)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 347)	Assignment operator.
◆ == (see page 347)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ AddGroup (see page 329)	Adds a group attribute field to the session.
◆ AddKeyMgmt (see page 329)	Adds a key management to the session.
◆ AddKeyMgmtParam (see page 330)	Adds a key management parameter to the session.
◆ AddMedia (see page 330)	Adds a level media.
◆ AddOtherAttribute (see page 330)	Adds an unknown attribute.
◆ AddPhone (see page 330)	Adds a phone field, i.e. "p=" field.
◆ AddTime (see page 331)	Adds a time field, i.e. "t=".
◆ FindGroupOfAMid (see page 331)	Returns the group in which the Mid is found.
◆ FindIdInMedias (see page 331)	Tells if the Id is present in a media and that media port is not 0.
◆ GetBandwidth (see page 331)	Gets the vector of Bandwidth field, i.e. "b=" field.
◆ GetConnectionData (see page 332)	Returns the connection data, i.e. "c=" field.
◆ GetDirection (see page 332)	Gets the direction attribute
◆ GetEmail (see page 332)	Gets the vector of Email field, i.e. "e=" field.
◆ GetEncryptionKey (see page 332)	Gets the Encryption key field, i.e. "k=" field.
◆ GetGroup (see page 333)	Gets the group attribute parameter at the specified index.
◆ GetIceOptions (see page 333)	Returns the ice-options attribute.
◆ GetIcePassword (see page 333)	Returns the ice-pwd attribute.
◆ GetIceUserFragment (see page 334)	Returns the ice-ufrag attribute.
◆ GetInformation (see page 334)	Gets the information field, i.e. "i=" field.
◆ GetKeyMgmt (see page 334)	Gets the key management attribute at the specified index.
◆ GetKeyMgmtParam (see page 334)	Gets the key management attribute parameter at the specified index.
◆ GetMedia (see page 335)	Returns the level media at the specified index.
◆ GetNbGroup (see page 335)	Gets the number of group attributes.
◆ GetNbKeyMgmt (see page 335)	Gets the number of key management attributes.
◆ GetNbKeyMgmtParam (see page 335)	Gets the number of key management parameters.
◆ GetNbMedias (see page 336)	Returns the number of level medias.
◆ GetNbOtherAttributes (see page 336)	Returns the number of unknown attributes.
◆ GetNbPhones (see page 336)	Returns the number of phone fields.
◆ GetNbTimes (see page 336)	Returns the number of time fields, i.e. "t=".
◆ GetOrigin (see page 336)	Returns the origin, i.e. "o=" field.
◆ GetOtherAttribute (see page 337)	Returns the unknown attribute at the specified index.
◆ GetPhone (see page 337)	Returns the phone field at the specified index.
◆ GetProtocolVersion (see page 337)	Returns the protocol version, i.e. "v=" field.
◆ GetSessionName (see page 338)	Returns the session name, i.e. "s=" field.
◆ GetTime (see page 338)	Returns the time at the specified index.
◆ GetUri (see page 338)	Gets the URI field, i.e. "u=" field.
◆ InsertMedia (see page 338)	Inserts a level media at the specified index.

◆ IsIceAttributePresent (see page 339)	Returns true if at least one ICE attribute is present at the session or media level.
◆ IsIceLite (see page 339)	The attribute a=ice-lite is present.
◆ IsInactive (see page 339)	The attribute "a=inactive" was present.
◆ IsRecvOnly (see page 339)	The attribute "a=recvonly" was present.
◆ IsSendOnly (see page 339)	The attribute "a=sendonly" was present.
◆ IsSendRecv (see page 340)	The attribute "a=sendrecv" was present.
◆ IsStreamPreferred (see page 340)	Returns true if the stream is the preferred one in the group it belongs.
◆ IsValidConnectionData (see page 340)	Returns if the connection data is valid or not, i.e. "c=" field.
◆ Parse (see page 340)	Parses all the needed information for this field.
◆ RemoveGroup (see page 341)	Removes a group attribute field from the session.
◆ RemoveKeyMgmt (see page 341)	Removes the key management from the session.
◆ RemoveKeyMgmtParam (see page 341)	Removes the key management parameter from the session.
◆ RemoveMedia (see page 341)	Removes the level media at the specified index.
◆ Reset (see page 342)	Resets all the data members.
◆ Serialize (see page 342)	Serializes the data contained in the level session.
◆ SetConnectionData (see page 342)	Sets the connection data, i.e. "c=" field.
◆ SetDirection (see page 343)	Sets the direction attribute
◆ SetEncryptionKey (see page 343)	Sets the Encryption key field, i.e. "k=" field.
◆ SetIceLite (see page 343)	Sets if an a=ice-lite attrbute is present.
◆ SetInactive (see page 343)	Sets an attribute to "inactive" value.
◆ SetInformation (see page 344)	Sets the information field, i.e. "i=" field.
◆ SetOrigin (see page 344)	Sets the origin, i.e. "o=" field.
◆ SetProtocolVersion (see page 344)	Sets the protocol version, i.e. "v=" field.
◆ SetRecvOnly (see page 344)	Sets an attribute to "recvonly" value.
◆ SetSendDirection (see page 344)	Sets whether or not to output the stream direction attributes when generating the packet.
◆ SetSendOnly (see page 345)	Sets an attribute to "sendonly" value.
◆ SetSendRecv (see page 345)	Sets an attribute to "sendrecv" value.
◆ SetSessionName (see page 345)	Sets the session name, i.e. "s=" field.
◆ SetT38BooleanEncoding (see page 345)	Sets the T.38 boolean encoding method of the T.38 streams.
◆ SetUri (see page 346)	Sets the URI field, i.e. "u=" field.
◆ UpdateGroupsIds (see page 346)	Updates identification in group field attributes to match accepted medias.
◆ Validate (see page 346)	Checks the validity of the parsed data.
◆ ValidateGrouping (see page 346)	Checks that the group field attributes are valid.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.48.1 - Constructors

10.1.48.1.1 - CSdpLevelSession

10.1.48.1.1.1 - CSdpLevelSession::CSdpLevelSession Constructor

Default constructor.

C++

```
CSdpLevelSession();
```

Description

Constructor

10.1.48.1.1.2 - CSdpLevelSession::CSdpLevelSession Constructor

Copy constructor.

C++

```
CSdpLevelSession(IN const CSdpLevelSession& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpLevelSession& rFrom`	The object to be copied.

Description

Copy constructor

10.1.48.2 - Destructors

10.1.48.2.1 - CSdpLevelSession::~CSdpLevelSession Destructor

Destructor.

C++

```
virtual ~CSdpLevelSession();
```

Description

Destructor

10.1.48.3 - Methods

10.1.48.3.1 - CSdpLevelSession::AddGroup Method

Adds a group attribute field to the session.

C++

```
void AddGroup(IN const CSdpFieldAttributeGroup& rGroup);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeGroup& rGroup`	The group attribute field to add to the session.

Description

Adds a "a=group:" field to the session. More than one can be added to the same session.

10.1.48.3.2 - CSdpLevelSession::AddKeyMgmt Method

Adds a key management to the session.

C++

```
void AddKeyMgmt(IN const CSdpFieldAttributeKeyMgmt& rKeyMgmt, IN const
CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH);
```

Parameters

Parameters	Description
`IN const CSdpFieldAttributeKeyMgmt& rKeyMgmt`	The key management attribute to add to the session.
`IN const CSdpFieldAttributeKeyMgmt::EKeyManagementAttributeRole eRole = CSdpFieldAttributeKeyMgmt::eBOTH`	The role that the key management attribute uses.

Description

Adds a key management attribute to the session.

Additionally, a CSdpKeyManagementParameter (see page 280) is added to every media. The parameter type is dependent on the key management type passed.

Note that any role previously set is overrided.

10.1.48.3.3 - CSdpLevelSession::AddKeyMgmtParam Method

Adds a key management parameter to the session.

C++

```
void AddKeyMgmtParam(IN const CSdpKeyManagementParameter& rKeyMgmtParam);
```

Parameters

Parameters	Description
`IN const CSdpKeyManagementParameter& rKeyMgmtParam`	The key management parameter.

Description

Adds a key management parameter to the session.

10.1.48.3.4 - CSdpLevelSession::AddMedia Method

Adds a level media.

C++

```
void AddMedia(IN const CSdpLevelMedia& rMedia);
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rMedia`	Reference to the CSdpLevelMedia (see page 288) to add.

Description

Adds a level media. More than one can be added to the same session.

10.1.48.3.5 - CSdpLevelSession::AddOtherAttribute Method

Adds an unknown attribute.

C++

```
void AddOtherAttribute(IN CSdpFieldAttributeOther& rOtherAttribute);
```

Parameters

Parameters	Description
`IN CSdpFieldAttributeOther& rOtherAttribute`	Reference to the CSdpFieldAttributeOther (see page 160) to add.

Description

Adds an unknown attribute. More than one can be added to the same session.

10.1.48.3.6 - CSdpLevelSession::AddPhone Method

Adds a phone field, i.e. "p=" field.

C++

```
void AddPhone(IN const CSdpFieldPhone& rPhone);
```

Parameters

Parameters	Description
`IN const CSdpFieldPhone& rPhone`	Reference to the CSdpFieldPhone (see page 251) to add.

Description

Adds a phone field, i.e. "p=" field. More than one can be added to the same session.

10.1.48.3.7 - CSdpLevelSession::AddTime Method

Adds a time field, i.e. "t=".

C++

```
void AddTime(IN CSdpFieldTime& rTime);
```

Parameters

Parameters	Description
`IN CSdpFieldTime& rTime`	Reference to the CSdpFieldTime (see page 263) to add.

Description

Adds a time field, i.e. "t=". More than one can be added to the same session.

10.1.48.3.8 - CSdpLevelSession::FindGroupOfAMid Method

Returns the group in which the Mid is found.

C++

```
const CSdpFieldAttributeGroup* FindGroupOfAMid(IN const CString& strMid) const;
```

Parameters

Parameters	Description
`IN const CString& strMid`	The Mid attribute from which to search.

Returns

The group where the Mid was found. NULL is returned if no group was found.

Description

This method searches in which group the Mid attribute is present.

10.1.48.3.9 - CSdpLevelSession::FindIdInMedias Method

Tells if the Id is present in a media and that media port is not 0.

C++

```
bool FindIdInMedias(IN const CString& strId, IN const bool bCheckDeactivatedMedia = false);
```

Parameters

Parameters	Description
`IN const CString& strId`	Identification tag to search.
`IN const bool bCheckDeactivatedMedia = false`	Tells if it should check inside a deactivated media or not.

Returns

True if the identification tag is present in a media and that the media has a port greater than 0.

Description

This method searches an identification tag from the media stream list of the provided capability manager.

10.1.48.3.10 - GetBandwidth

10.1.48.3.10.1 - CSdpLevelSession::GetBandwidth Method

Gets the vector of Bandwidth field, i.e. "b=" field.

C++

```
CVector<CString>& GetBandwidth();
```

```
const CVector<CString>& GetBandwidth() const;
```

Returns

Reference to the bandwidth vector.

Description

Gets the vector of Bandwidth field, i.e. "b=" field.

10.1.48.3.11 - GetConnectionData

10.1.48.3.11.1 - CSdpLevelSession::GetConnectionData Method

Returns the connection data, i.e. "c=" field.

C++

```
CSdpFieldConnectionData& GetConnectionData();
const CSdpFieldConnectionData& GetConnectionData() const;
```

Returns

Reference to the CSdpFieldConnectionData (see page 228).

Description

Returns the connection data, i.e. "c=" field.

10.1.48.3.12 - CSdpLevelSession::GetDirection Method

Gets the direction attribute

C++

```
EAttributeType GetDirection() const;
```

Returns

The direction attribute.

Description

Gets the direction attribute.

See Also

IsSendOnly (see page 339), IsRecvOnly (see page 339), IsSendRecv (see page 340), IsInactive (see page 339)

10.1.48.3.13 - GetEmail

10.1.48.3.13.1 - CSdpLevelSession::GetEmail Method

Gets the vector of Email field, i.e. "e=" field.

C++

```
CVector<CString>& GetEmail();
const CVector<CString>& GetEmail() const;
```

Returns

Reference to the vector of email field.

Description

Gets the vector of Email field, i.e. "e=" field.

10.1.48.3.14 - CSdpLevelSession::GetEncryptionKey Method

Gets the Encryption key field, i.e. "k=" field.

C++

```
const CString& GetEncryptionKey() const;
```

Returns

Reference to the CString encryption key.

Description

Gets the Encryption key field, i.e. "k=" field.

10.1.48.3.15 - GetGroup

10.1.48.3.15.1 - CSdpLevelSession::GetGroup Method

Gets the group attribute parameter at the specified index.

C++

```
CSdpFieldAttributeGroup* GetGroup(IN uint16_t uIndex);
const CSdpFieldAttributeGroup* GetGroup(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the attribute to retrieve.

Returns

The specified group from the session. NULL is returned if the index is greater than or equal to the number of groups.

Description

Gets the specified group from the session. uIndex must be smaller than the actual number of groups. NULL is returned otherwise.

10.1.48.3.16 - GetIceOptions

10.1.48.3.16.1 - CSdpLevelSession::GetIceOptions Method

Returns the ice-options attribute.

C++

```
CSdpFieldAttributeIceOptions& GetIceOptions();
const CSdpFieldAttributeIceOptions& GetIceOptions() const;
```

Returns

A reference to CSdpFieldAttributeIceOptions (see page 114).

Description

Returns a reference to the ICE options.

10.1.48.3.17 - GetIcePassword

10.1.48.3.17.1 - CSdpLevelSession::GetIcePassword Method

Returns the ice-pwd attribute.

C++

```
CSdpFieldAttributeIcePwd& GetIcePassword();
const CSdpFieldAttributeIcePwd& GetIcePassword() const;
```

Returns

A reference to CSdpFieldAttributeIcePwd (see page 118).

Description

Returns a reference to the ICE password.

10.1.48.3.18 - GetIceUserFragment

10.1.48.3.18.1 - CSdpLevelSession::GetIceUserFragment Method

Returns the ice-ufrag attribute.

C++

```
CSdpFieldAttributeIceUserFrag& GetIceUserFragment();
const CSdpFieldAttributeIceUserFrag& GetIceUserFragment() const;
```

Returns

A reference to CSdpFieldAttributeIceUserFrag (see page 131).

Description

Returns a reference to the ICE user fragment.

10.1.48.3.19 - CSdpLevelSession::GetInformation Method

Gets the information field, i.e. "i=" field.

C++

```
const CString& GetInformation() const;
```

Returns

CString reference to the information field.

Description

Gets the information field, i.e. "i=" field.

10.1.48.3.20 - GetKeyMgmt

10.1.48.3.20.1 - CSdpLevelSession::GetKeyMgmt Method

Gets the key management attribute at the specified index.

C++

```
CSdpFieldAttributeKeyMgmt& GetKeyMgmt(IN uint16_t uIndex);
const CSdpFieldAttributeKeyMgmt& GetKeyMgmt(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the attribute to retrieve.

Description

Gets the specified key management attribute from the session.

10.1.48.3.21 - GetKeyMgmtParam

10.1.48.3.21.1 - CSdpLevelSession::GetKeyMgmtParam Method

Gets the key management attribute parameter at the specified index.

C++

```
CSdpKeyManagementParameter& GetKeyMgmtParam(IN uint16_t uIndex);
const CSdpKeyManagementParameter& GetKeyMgmtParam(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the attribute parameter to retrieve.

Description

Gets the specified key management parameter from the session.

10.1.48.3.22 - GetMedia

10.1.48.3.22.1 - CSdpLevelSession::GetMedia Method

Returns the level media at the specified index.

C++

```
CSdpLevelMedia& GetMedia(IN uint16_t uIndex);
const CSdpLevelMedia& GetMedia(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the media to retrieve.

Returns

Reference to the requested CSdpLevelMedia (see page 288).

Description

Gets the level media at the specified index.

10.1.48.3.23 - CSdpLevelSession::GetNbGroup Method

Gets the number of group attributes.

C++

```
uint32_t GetNbGroup() const;
```

Returns

The number of group attributes in the session.

Description

Returns the number of group attributes in the session.

10.1.48.3.24 - CSdpLevelSession::GetNbKeyMgmt Method

Gets the number of key management attributes.

C++

```
uint32_t GetNbKeyMgmt() const;
```

Returns

The number of key management attributes in the session.

Description

Gets the number of key management attributes in the session.

10.1.48.3.25 - CSdpLevelSession::GetNbKeyMgmtParam Method

Gets the number of key management parameters.

C++

```
uint32_t GetNbKeyMgmtParam() const;
```

Returns

The number of key management parameters in the session.

Description

Gets the number of key management parameters in the session.

10.1.48.3.26 - CSdpLevelSession::GetNbMedias Method

Returns the number of level medias.

C++

```
uint32_t GetNbMedias() const;
```

Returns

The number of level medias.

Description

Gets the number of level medias.

10.1.48.3.27 - CSdpLevelSession::GetNbOtherAttributes Method

Returns the number of unknown attributes.

C++

```
uint32_t GetNbOtherAttributes() const;
```

Returns

Number of unknown attributes.

Description

Returns the number of unknown attributes.

10.1.48.3.28 - CSdpLevelSession::GetNbPhones Method

Returns the number of phone fields.

C++

```
uint32_t GetNbPhones() const;
```

Returns

Number of phone fields.

Description

Returns the number of phone fields, i.e. "p=".

10.1.48.3.29 - CSdpLevelSession::GetNbTimes Method

Returns the number of time fields, i.e. "t=".

C++

```
uint32_t GetNbTimes() const;
```

Returns

Number of time fields.

Description

Returns the number of time fields, i.e. "t=".

10.1.48.3.30 - GetOrigin

10.1.48.3.30.1 - CSdpLevelSession::GetOrigin Method

Returns the origin, i.e. "o=" field.

C++

```
CSdpFieldOrigin& GetOrigin();
const CSdpFieldOrigin& GetOrigin() const;
```

Returns

Reference to the requested CSdpFieldOrigin (see page 243).

Description

Returns the origin, i.e. "o=" field.

10.1.48.3.31 - GetOtherAttribute

10.1.48.3.31.1 - CSdpLevelSession::GetOtherAttribute Method

Returns the unknown attribute at the specified index.

C++

```
CSdpFieldAttributeOther& GetOtherAttribute(IN uint16_t uIndex);
const CSdpFieldAttributeOther& GetOtherAttribute(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the attribute to retrieve.

Returns

Reference to the requested CSdpFieldAttributeOther (see page 160).

Description

Gets the unknown attribute at the specified index.

10.1.48.3.32 - CSdpLevelSession::GetPhone Method

Returns the phone field at the specified index.

C++

```
const CSdpFieldPhone& GetPhone(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the requested phone field.

Returns

Reference to the CSdpFieldPhone (see page 251) requested

Description

Returns the phone field at the specified index.

10.1.48.3.33 - GetProtocolVersion

10.1.48.3.33.1 - CSdpLevelSession::GetProtocolVersion Method

Returns the protocol version, i.e. "v=" field.

C++

```
CSdpFieldProtocolVersion& GetProtocolVersion();
const CSdpFieldProtocolVersion& GetProtocolVersion() const;
```

Returns

Reference to the requested CSdpFieldProtocolVersion (see page 255).

Description

Returns the protocol version, i.e. "v=" field.

10.1.48.3.34 - GetSessionName

10.1.48.3.34.1 - CSdpLevelSession::GetSessionName Method

Returns the session name, i.e. "s=" field.

C++

```
CSdpFieldSessionName& GetSessionName();
const CSdpFieldSessionName& GetSessionName() const;
```

Returns

Reference to the requested CSdpFieldSessionName (see page 259).

Description

Returns the session name, i.e. "s=" field.

10.1.48.3.35 - GetTime

10.1.48.3.35.1 - CSdpLevelSession::GetTime Method

Returns the time at the specified index.

C++

```
CSdpFieldTime& GetTime(IN uint16_t uIndex);
const CSdpFieldTime& GetTime(IN uint16_t uIndex) const;
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	index of the times vector.

Returns

Reference to the CSdpFieldTime (see page 263).

Description

Returns the time at the specified index.

10.1.48.3.36 - CSdpLevelSession::GetUri Method

Gets the URI field, i.e. "u=" field.

C++

```
const CString& GetUri() const;
```

Returns

CString reference to the URI field.

Description

Gets the URI field, i.e. "u=" field.

10.1.48.3.37 - CSdpLevelSession::InsertMedia Method

Inserts a level media at the specified index.

C++

```
void InsertMedia(IN uint32_t uIndex, IN const CSdpLevelMedia& rMedia);
```

Parameters

Parameters	Description
`IN uint32_t uIndex`	Index of where to insert the media.
`IN const CSdpLevelMedia& rMedia`	Reference of the CSdpLevelMedia (see page 288) to insert.

Description

Inserts a level media at the specified index.

10.1.48.3.38 - CSdpLevelSession::IsIceAttributePresent Method

Returns true if at least one ICE attribute is present at the session or media level.

C++

```
bool IsIceAttributePresent() const;
```

Returns

true if the session or medias contain at least one ICE related attributes.

Description

Returns true if the media contains at least one of the ICE attributes: ice-lite, ice-ufrag, ice-pwd, ice-options, candidate or remote-candidates.

10.1.48.3.39 - CSdpLevelSession::IsIceLite Method

The attribute a=ice-lite is present.

C++

```
bool IsIceLite() const;
```

Returns

- true: If an a=ice-lite attribute is present.
- false: Otherwise.

Description

Returns whether or not an a=ice-lite attribute is present.

10.1.48.3.40 - CSdpLevelSession::IsInactive Method

The attribute "a=inactive" was present.

C++

```
bool IsInactive() const;
```

Returns

- True: The attribute "a=inactive" is present.
- False: The attribute "a=inactive" is not present.

Description

Verifies if the attribute "a=inactive" is present.

10.1.48.3.41 - CSdpLevelSession::IsRecvOnly Method

The attribute "a=recvonly" was present.

C++

```
bool IsRecvOnly() const;
```

Returns

- True: The attribute "a=recvonly" is present.
- False: The attribute "a=recvonly" is not present.

Description

Verifies if the attribute "a=recvonly" is present.

10.1.48.3.42 - CSdpLevelSession::IsSendOnly Method

The attribute "a=sendonly" was present.

C++

```
bool IsSendOnly() const;
```

Returns

- True: The attribute "a=sendonly" is present.
- False: The attribute "a=sendonly" is not present.

Description

Verifies if the attribute "a=sendonly" is present.

10.1.48.3.43 - CSdpLevelSession::IsSendRecv Method

The attribute "a=sendrecv" was present.

C++

```
bool IsSendRecv() const;
```

Returns

- True: The attribute "a=sendrecv" is present.
- False: The attribute "a=sendrecv" is not present.

Description

Verifies if the attribute "a=sendrecv" is present.

10.1.48.3.44 - CSdpLevelSession::IsStreamPreferred Method

Returns true if the stream is the preferred one in the group it belongs.

C++

```
bool IsStreamPreferred(IN const CSdpLevelMedia& rOfferStream) const;
```

Parameters

Parameters	Description
`IN const CSdpLevelMedia& rOfferStream`	The media stream to check.

Returns

True if the stream is the preferred one in its group or if there is no group.

Description

This method checks if the stream belongs to a group, if so, it returns true. If no group is present, the stream is automatically the preferred one.

10.1.48.3.45 - CSdpLevelSession::IsValidConnectionData Method

Returns if the connection data is valid or not, i.e. "c=" field.

C++

```
bool IsValidConnectionData() const;
```

Returns

- True: Tthe connection data is valid
- False: The connection data is not valid

Description

Verifies whether or not the connection data is valid, i.e. "c=" field.

10.1.48.3.46 - CSdpLevelSession::Parse Method

Parses all the needed information for this field.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	result value.

Returns

Value used to control the parsing.

Description

Parses all the needed information for this session.

10.1.48.3.47 - CSdpLevelSession::RemoveGroup Method

Removes a group attribute field from the session.

C++

```
void RemoveGroup(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the attribute to remove.

Description

Removes a "a=group:" field from the session.

10.1.48.3.48 - CSdpLevelSession::RemoveKeyMgmt Method

Removes the key management from the session.

C++

```
void RemoveKeyMgmt(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	The index of the key management attribute to remove.

Description

Removes the requested key management attribute from the session.

10.1.48.3.49 - CSdpLevelSession::RemoveKeyMgmtParam Method

Removes the key management parameter from the session.

C++

```
void RemoveKeyMgmtParam(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index of the key management parameter to remove.

Description

Removes the key management parameter at the specified index.

10.1.48.3.50 - CSdpLevelSession::RemoveMedia Method

Removes the level media at the specified index.

C++

```
void RemoveMedia(IN uint16_t uIndex);
```

Parameters

Parameters	Description
`IN uint16_t uIndex`	Index to remove from the media vector.

Description

Removes the level media at the specified index.

10.1.48.3.51 - CSdpLevelSession::Reset Method

Resets all the data members.

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 340).

10.1.48.3.52 - CSdpLevelSession::Serialize Method

Serializes the data contained in the level session.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

draft-ietf-mmusic-sdp-new-24.txt ABNF:

```
session-description = proto-version
                      origin-field
                      session-name-field
                      information-field
                      uri-field
                      email-fields
                      phone-fields
                      connection-field
                      bandwidth-fields
                      time-fields
                      key-field
                      attribute-fields
                      media-descriptions
```

10.1.48.3.53 - CSdpLevelSession::SetConnectionData Method

Sets the connection data, i.e. "c=" field.

C++

```
void SetConnectionData(IN const CSdpFieldConnectionData& rConnectionData);
```

Parameters

Parameters	Description
`IN const CSdpFieldConnectionData& rConnectionData`	Reference to the CSdpFieldConnectionData (see page 228) to set.

Description

Sets the connection data, i.e. "c=" field.

10.1.48.3.54 - CSdpLevelSession::SetDirection Method

Sets the direction attribute

C++

```
void SetDirection(IN EAttributeType eDirection);
```

Parameters

Parameters	Description
`IN EAttributeType eDirection`	Direction attribute (EAttributeType)

Description

Sets the direction attribute

See Also

SetSendOnly (see page 345), SetRecvOnly (see page 344), SetSendRecv (see page 345), SetInactive (see page 343)

10.1.48.3.55 - CSdpLevelSession::SetEncryptionKey Method

Sets the Encryption key field, i.e. "k=" field.

C++

```
void SetEncryptionKey(IN const char* szEncryptionKey);
```

Parameters

Parameters	Description
`IN const char* szEncryptionKey`	The Encryption key field content to set.

Description

Sets the Encryption key field, i.e. "k=" field.

10.1.48.3.56 - CSdpLevelSession::SetIceLite Method

Sets if an a=ice-lite attrbute is present.

C++

```
void SetIceLite(IN bool bIsIceLite);
```

Parameters

Parameters	Description
`IN bool bIsIceLite`	Whether or not an a=ice-lite attribute is present.

Description

Sets whether or not an a=ice-lite attribute is present.

10.1.48.3.57 - CSdpLevelSession::SetInactive Method

Sets an attribute to "inactive" value.

C++

```
void SetInactive(IN bool bInactive);
```

Parameters

Parameters	Description
`IN bool bInactive`	• True to set the inactive value. • False otherwise.

Description

Sets an attribute to "inactive" value.

10.1.48.3.58 - CSdpLevelSession::SetInformation Method

Sets the information field, i.e. "i=" field.

C++

```
void SetInformation(IN const char* szInformation);
```

Parameters

Parameters	Description
`IN const char* szInformation`	The information field content to set.

Description

Sets the information field, i.e. "i=" field.

10.1.48.3.59 - CSdpLevelSession::SetOrigin Method

Sets the origin, i.e. "o=" field.

C++

```
void SetOrigin(IN const CSdpFieldOrigin& rOrigin);
```

Parameters

Parameters	Description
`IN const CSdpFieldOrigin& rOrigin`	Reference to the CSdpFieldOrigin (see page 243) to set.

Description

Sets the origin, i.e. "o=" field.

10.1.48.3.60 - CSdpLevelSession::SetProtocolVersion Method

Sets the protocol version, i.e. "v=" field.

C++

```
void SetProtocolVersion(IN CSdpFieldProtocolVersion& rProtocolVersion);
```

Parameters

Parameters	Description
`IN CSdpFieldProtocolVersion& rProtocolVersion`	Reference to the CSdpFieldProtocolVersion (see page 255) to set.

Description

Sets the protocol version, i.e. "v=" field.

10.1.48.3.61 - CSdpLevelSession::SetRecvOnly Method

Sets an attribute to "recvonly" value.

C++

```
void SetRecvOnly(IN bool bRecvOnly);
```

Parameters

Parameters	Description
`IN bool bRecvOnly`	• True to set the recvonly value. • False otherwise.

Description

Sets an attribute to "recvonly" value.

10.1.48.3.62 - CSdpLevelSession::SetSendDirection Method

Sets whether or not to output the stream direction attributes when generating the packet.

C++

```
void SetSendDirection(IN bool bSend);
```

Parameters

Parameters	Description
`IN bool bSend`	• True to output the stream direction to all medias.. • False otherwise.

Description

Sets whether or not to output the stream direction attributes when generating the packet.

10.1.48.3.63 - CSdpLevelSession::SetSendOnly Method

Sets an attribute to "sendonly" value.

C++

```
void SetSendOnly(IN bool bSendOnly);
```

Parameters

Parameters	Description
`IN bool bSendOnly`	• True to set the sendonly value. • False otherwise.

Description

Sets an attribute to "sendonly" value.

10.1.48.3.64 - CSdpLevelSession::SetSendRecv Method

Sets an attribute to "sendrecv" value.

C++

```
void SetSendRecv(IN bool bSendRecv);
```

Parameters

Parameters	Description
`IN bool bSendRecv`	• True to set the sendrecv value. • False otherwise.

Description

Sets an attribute to "sendrecv" value.

10.1.48.3.65 - CSdpLevelSession::SetSessionName Method

Sets the session name, i.e. "s=" field.

C++

```
void SetSessionName(IN CSdpFieldSessionName& rSessionName);
```

Parameters

Parameters	Description
`rOrigin`	Reference to the CSdpFieldSessionName (see page 259) to set.

Description

Sets the session name, i.e. "s=" field.

10.1.48.3.66 - CSdpLevelSession::SetT38BooleanEncoding Method

Sets the T.38 boolean encoding method of the T.38 streams.

C++

```
void SetT38BooleanEncoding(bool bT38ImplicitEncoding);
```

Parameters

Parameters	Description
`bool bT38ImplicitEncoding`	Indicates if the encoding method is the implicit method (true) or the explicit method (false).

Description

Sets the T.38 boolean encoding method of the T.38 streams.

10.1.48.3.67 - CSdpLevelSession::SetUri Method

Sets the URI field, i.e. "u=" field.

C++

```
void SetUri(IN const char* szUri);
```

Parameters

Parameters	Description
`IN const char* szUri`	The URI field content to set.

Description

Sets the URI field, i.e. "u=" field.

10.1.48.3.68 - CSdpLevelSession::UpdateGroupsIds Method

Updates identification in group field attributes to match accepted medias.

C++

```
void UpdateGroupsIds();
```

Description

This method updates the list of identification tags present in every group. If a media has its port set to 0, the tag is removed from the groups.

10.1.48.3.69 - CSdpLevelSession::Validate Method

Checks the validity of the parsed data.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.48.3.70 - CSdpLevelSession::ValidateGrouping Method

Checks that the group field attributes are valid.

C++

```
void ValidateGrouping();
```

Description

This method validates that the group field attributes present in the offer are supported. If not, they are removed.

10.1.48.4 - Operators

10.1.48.4.1 - CSdpLevelSession::= Operator

Assignment operator.

C++

```
CSdpLevelSession& operator =(IN const CSdpLevelSession& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpLevelSession& rFrom`	The right operand of the assignment (to copy in *this).

Returns

A reference to this, to enable concatenation.

Description

Assignment operator

10.1.48.4.2 - CSdpLevelSession::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpLevelSession& rFrom) const;
```

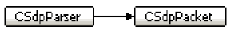
Description

Equality operator

10.1.49 - CSdpPacket Class

This class implements an abstraction of a SDP packet.

Class Hierarchy



C++

```
class CSdpPacket : public CSdpParser;
```

Description

This class is an abstraction of a SDP packet. The parsing of this class corresponds to the BNF described in RFC 2327.

RFC 2327 BNF:

sdp-packet = level-session

How to use this class:

- Steps to parse a SDP packet: 1) Create a CSdpPacket object 2) Call Parse (see page 349)() to read the data from a buffer 3) Check that rres == resS_OK to make sure no parsing error has occurred 4) Call IsValid (see page 353)() to make sure all the data has been correctly read 5) Call all the GetXXX() functions needed to get the desired information 6) Repeat steps 2, 3, 4 and 5 as needed, but be aware that it is your responsibility to call Reset (see page 350) between consecutive calls to Parse (see page 349) on the same object

- Steps to generate a SDP packet: 1) Create a CSdpPacket object, a CSdpLevelSession (see page 326) one, and CSdpLevelMedia (see page 288)(s) if needed 2) Call all the SetXXX() and AddXXX() functions needed to set the desired information 3) Call Validate (see page 350)() to make sure the packet at least contains the minimum fields and that all the fields are valid 4) Call Serialize (see page 350)() to put the data in a CBlob object

Location

SdpParser/CSdpPacket.h

See Also

CSdpLevelSession (see page 326)

Constructors

Constructor	Description
◆ CSdpPacket (see page 349)	Default constructor.

CSdpParser Class

CSdpParser Class	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpPacket (see page 349)	Destructor.

CSdpParser Class

CSdpParser Class	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 351)	Assignment operator.
◆ == (see page 351)	Comparison operator.

CSdpParser Class

CSdpParser Class	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ GetSession (see page 349)	Gets the session.
◆ Parse (see page 349)	Parses all needed information for this packet.
◆ Reset (see page 350)	Needed in order to recall Parse (see page 349).
◆ Serialize (see page 350)	Generates the data blob from the data members.
◆ SetSession (see page 350)	Sets the session.
◆ Validate (see page 350)	Validates the parsed data for this packet.

CSdpParser Class

CSdpParser Class	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.49.1 - Constructors

10.1.49.1.1 - CSdpPacket

10.1.49.1.1.1 - CSdpPacket::CSdpPacket Constructor

Default constructor.

C++

```
CSdpPacket();
```

Description

Constructor

10.1.49.1.1.2 - CSdpPacket::CSdpPacket Constructor

Copy constructor.

C++

```
CSdpPacket(IN const CSdpPacket& rFrom);
```

Description

Copy Constructor

10.1.49.2 - Destructors

10.1.49.2.1 - CSdpPacket::~CSdpPacket Destructor

Destructor.

C++

```
virtual ~CSdpPacket();
```

Description

Destructor

10.1.49.3 - Methods

10.1.49.3.1 - GetSession

10.1.49.3.1.1 - CSdpPacket::GetSession Method

Gets the session.

C++

```
CSdpLevelSession& GetSession();
const CSdpLevelSession& GetSession() const;
```

Returns

The session level.

Description

Returns the session level.

10.1.49.3.2 - CSdpPacket::Parse Method

Parses all needed information for this packet.

C++

```
EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres);
```

Parameters

Parameters	Description
`INOUT const char*& rpszStartPosition`	Pointer to the data to be parsed.
`OUT mxt_result& rres`	Result value.

Returns

Value used to control the parsing. It is not very useful, but it may be interesting to know that it can take the following values, if rres does not report an error: eOK_CONTINUE: Some data remains after the parsed data. eOK_NULL: A NULL character has been found after the parsed data. If rres reports an error, the return value can take any value in EParserResult and is absolutely useless anyway.

Description

Parses all the needed information for this packet.

WARNINGS: It is the responsibility of the caller to call Reset (see page 350)() between consecutive calls to Parse on the same object.

10.1.49.3.3 - CSdpPacket::Reset Method

Needed in order to recall Parse (see page 349).

C++

```
void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 349).

10.1.49.3.4 - CSdpPacket::Serialize Method

Generates the data blob from the data members.

C++

```
void Serialize(INOUT CBlob& rBlob) const;
```

Parameters

Parameters	Description
`INOUT CBlob& rBlob`	The CBlob object where the data is stored.

Description

Generates the data blob from the data members.

10.1.49.3.5 - CSdpPacket::SetSession Method

Sets the session.

C++

```
void SetSession(IN CSdpLevelSession& rSession);
```

Parameters

Parameters	Description
`IN CSdpLevelSession& rSession`	The session level to set.

Description

Sets the session level.

10.1.49.3.6 - CSdpPacket::Validate Method

Validates the parsed data for this packet.

C++

```
bool Validate();
```

Returns

- True: the attribute is valid.
- False: the attribute is invalid.

Description

Sets the value of the flag 'm_bIsValid' by checking the validity of the parsed data and returns this value.

10.1.49.4 - Operators

10.1.49.4.1 - CSdpPacket::= Operator

Assignment operator.

C++

```
CSdpPacket& operator =(IN const CSdpPacket& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpPacket& rFrom`	The object to be copied.

Description

Member data values are assigned to the LHS object from the member data values of the RHS object.

10.1.49.4.2 - CSdpPacket::== Operator

Comparison operator.

C++

```
bool operator ==(IN const CSdpPacket& rFrom) const;
```

Returns

true if both attributes contain the same user fragment.

Description

Comparison operator

10.1.50 - CSdpParser Class

This class implements the abstract base class for all the other SDP Parser classes.

Class Hierarchy

C++

```
class CSdpParser;
```

Description

This is the abstract base class for all the other SDP Parser classes. It contains definitions of pure virtual functions to be defined in the concrete classes. It also contains some useful parsing functions.

For every field, the parser follows what is described in RFC 2327. The enumeration EParserType describes in more details what is parsed.

SDP session descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attribute names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set.

The fields in a SDP Packet contain one or several tokens which this base class offers methods to parse properly.

CSdpParser::GetLine
CSdpParser::GetToken
CSdpParser::GetSubToken

These methods and the Parse (see page 353) method of every Child return a type of the enumeration EParserResult.

Location

SdpParser/CSdpParser.h

Constructors

Constructor	Description
◆ CSdpParser (see page 352)	Default constructor.

Legend

◆	Method

Destructors

Destructor	Description
◆ V ~CSdpParser (see page 353)	Destructor.

Legend

◆	Method
V	virtual

Operators

Operator	Description
◆ = (see page 354)	Assignment operator.

Legend

◆	Method

Methods

Method	Description
◆ IsValid (see page 353)	Returns true if the data was parsed successfully.
◆ A Parse (see page 353)	Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.
◆ V Reset (see page 353)	Resets the data in the parser.
◆ A Validate (see page 353)	Validates the parsed data.

Legend

◆	Method
A	abstract
V	virtual

10.1.50.1 - Constructors

10.1.50.1.1 - CSdpParser

10.1.50.1.1.1 - CSdpParser::CSdpParser Constructor

Default constructor.

C++

```
CSdpParser();
```

Description

Constructor

10.1.50.1.1.2 - CSdpParser::CSdpParser Constructor

Copy constructor.

C++

```
CSdpParser(IN const CSdpParser& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpParser& rFrom`	The CSdpParser to be copied.

Description

Copy constructor.

10.1.50.2 - Destructors

10.1.50.2.1 - CSdpParser::~CSdpParser Destructor

Destructor.

C++

```
virtual ~CSdpParser();
```

Description

Destructor

10.1.50.3 - Methods

10.1.50.3.1 - CSdpParser::IsValid Method

Returns true if the data was parsed successfully.

C++

```
bool IsValid() const;
```

Returns

True if the data was parsed successfully.

Description

Returns whether or not the parsing succeeded.

10.1.50.3.2 - CSdpParser::Parse Method

Parses the parameters list beginning at rpszStartPosition. Can return any type of EParserResult.

C++

```
virtual EParserResult Parse(INOUT const char*& rpszStartPosition, OUT mxt_result& rres) = 0;
```

10.1.50.3.3 - CSdpParser::Reset Method

Resets the data in the parser.

C++

```
virtual void Reset();
```

Description

Resets all the data members, to be ready for another call to Parse (see page 353).

10.1.50.3.4 - CSdpParser::Validate Method

Validates the parsed data.

C++

```
virtual bool Validate() = 0;
```

10.1.50.4 - Operators

10.1.50.4.1 - CSdpParser::= Operator

Assignment operator.

C++

```
CSdpParser& operator =(IN const CSdpParser& rFrom);
```

Parameters

Parameters	Description
`IN const CSdpParser& rFrom`	The CSdpParser (see page 351) to be copied.

Description

Assignment operator.

10.1.50.5 - Friends

10.1.50.5.1 - friend class CSdpCapabilitiesMgr Friend

Friend class used by CSdpParser (see page 351)

C++

```
friend class CSdpCapabilitiesMgr;
```

Index