Avaya Aura® Session Manager Call Detail Recording Interface

Abstract

The purpose of this document is to provide the technical interface details necessary to design and build a product which is capable of successfully collecting and managing the CDR data files produced by the Avaya Aura® Session Manager.

Disclaimer: The information contained in this document is current and is thought to be correct at the time of publication. Please check the release documentation on support.avaya.com for the most up to date details.

TABLE OF CONTENTS

TABLE OF TABLES

1. INTRODUCTION

This document is intended to provide the technical details necessary to successfully design and build a product capable of retrieving and managing the CDR data files produced by Avaya Aura® Session Manager.

Release 6.3.4 supports multiple CDR formats selectable through administration.

Other topics covered in this document include:

- A description of Avaya Aura® Session Manager CDR interface architecture
- The required security provisions
- The CDR data file naming convention and structure used by Avaya Aura® Session Manager CDR
- The supported data transport protocols
- A description of the provisions employed to delete the CDR data files
- An example scenario describing the provisions, procedures, and concerns associated with the Avaya Aura® Session Manager CDR

Note to the Reader: Unless specified otherwise, throughout this document, the terms "server" and "Session Manager" are intended to refer to Avaya Aura® Session Manager.

1.1 INTENDED AUDIENCE

The document is intended for a community of companies and organizations that design, build, market, and support Session Manager CDR adjunct systems.

1.2 TERMINOLOGY AND ACRONYMS

Table 1 summarizes some of the acronyms and terminologies used in this document.

Table 1 - Terminology and Acronyms

TERM	DESCRIPTION
ASBCE	Avaya Session Border Controller for Enterprise
CDR	Call Detail Record or Call Detail Recording
CM	Avaya Aura® Communication Manager
GSID	Global Session ID
MA-UUI	No message-associated user-to-user signaling
SFTP	Secure File Transfer Protocol or SSH File Transfer Protocol
SM	Avaya Aura® Session Manager

2. INTERFACE ARCHITECTURE HIGHLIGHTS

2.1 SESSION MANAGER CDR

The architecture of this feature centers on saving CDR records on the local hard drive of the server instead of transmitting CDR call files over an IP connection to the CDR adjunct.

When the Session Manager CDR feature is properly administered on a server, Session Manager saves the CDR records on the hard drive of its local server. Periodically, the CDR adjunct must log on to each of the Session Manager's servers and retrieve all the available CDR data files.

The CDR adjunct logs into the server through a special login and password that the server administrator creates. The special login account is a member of the “CDR_User” group. For security reasons, the special login of CDR adjunct is only given access to the directory where the CDR records are stored.

The CDR format utilized in the first release of Session Manager is compatible with existing CDR adjuncts connecting to the Avaya Aura® Communication Manager servers implementing the “Survivable CDR” feature. The Survivable CDR feature utilizes the “Unformatted” record format of CM Release 4.0 and higher.

A CDR adjunct used in a configuration including multiple Session Managers must simultaneously receive and process the formats generated by each Session Manager in the configuration. A customer can have a set of Session Managers running on two different releases simultaneously. Session Managers may generate different CDR output formats depending on the releases. CDR adjunct must be capable of processing the different formats.

2.2 CDR RECORD FORMAT

Before SM 6.3.4, there was a single CDR format for a given release. Starting with SM 6.3.4, three different formats are supported. You can select the format from the Session Manager instance page on the System Manager web console. The three formats are called ‘standard flat file’, ‘enhanced flat file’, and ‘enhanced XML’. The standard flat file format is identical to the format in SM 6.3.2. The enhanced flat file format contains newly supported fields for call usage, codec, and tenant id and uses different field widths than the standard flat file format. The enhanced XML format supports the same new fields as the enhanced flat file format.

Session Manager produced CDR data files may contain two types of data records:

- date records
- call detail records

All CDR formats produced before SM 6.3.4 contained both types of data records. From SM 6.3.4, the standard flat file and enhanced flat file formats contain date records and call detail records. The enhanced XML format contains only call detail records because the call detail records contain the time and date of the call, so no separate date record is needed.

All data contained in the CDR data files is stored in standard ASCII characters.

Date Record

The date record is a timestamp in the CDR data file. The date record indicates to the CDR processing application that every call record that follows in the file is referred to the new date. The date record is used to synchronize the CDR adjunct’s time and date processing with the starting time and date of the information in the CDR data file. Multiple date records can be put in a single CDR data file, even if no date change occurred. The SM CDR system provides date information to the CDR data file:

- once a day at midnight
- when the date or time is manually changed on the server
- when a new CDR data file is created

To avoid inconsistencies between the date record and the call records following it, it is a common practice of the server to generate a new CDR data file just before midnight with all of the completed calls at that time. The new CDR data file is generated regardless of the set file creation schedule.
In the following table, the call records contain the time the call ended and the duration. Therefore, the remaining calls ending after midnight will be reported in the next CDR file. The next CDR file

uses a date record with the new date. Date records are 13 characters. The format of the Date record is as follows:

Table 2 - Date Record Format

Position	Description
1-2	Hour (leading 0s added if needed)
3	Colon (":")
4-5	Minute (leading 0s added if needed)
6	Blank
7-8	Month (leading 0s added if needed)
9	Slash ("/")
10-11	Day (leading 0s added if needed)
12	Carriage Return
13	Line Feed

For example, a CDR date record can be:

21:32 04/16<CR><LF>

Call Detail Record

Originally in SM CDR, each call detail record contained 106 characters. The format was as follows:

Table 3 - Call Detail Record Data Format before SM 6.1

Position	Description
1-2	Time of day-hours
3-4	Time of day-minutes
5	Duration-hours
6-7	Duration-minutes
8	Duration-tenths of minutes
9	Condition Code
10-17	Space
18-32	Dialed number
33-42	Calling number
43-48	Space
49-55	Terminating SIP Entity
56-57	Space
58-64	Originating SIP Entity
65-73	Space
74	Feature Flag
75-92	Space
93	Bearer Capability Class
94	MA-UUI
95	Resource Flag
96-104	Space
105	Carriage Return
106	Line Feed

Beginning with SM 6.1, each call detail record was expanded to 108 characters to accommodate the new Bandwidth field. The format is as follows:

Table 4 - Call Detail Record Data Format in SM 6.1 and SM 6.2

Position	Description
1-2	Time of day-hours
3-4	Time of day-minutes
5	Duration-hours
6-7	Duration-minutes
8	Duration-tenths of minutes
9	Condition code
10-17	Space
18-32	Dialed number
33-42	Calling number
43-48	Space
49-55	Terminating SIP Entity
56-57	Space
58-64	Originating SIP Entity
65-73	Space
74	Feature flag
75-92	Space
93	BCC
94	MA-UUI
95	Resource flag
96-104	Space
105-106	Bandwidth
107	Carriage return
108	Line feed

Table 5 - Call Detail Record Data Format in SM 6.3.x before 6.3.4 and Standard Flat File Format in SM 6.3.4

Position	Description
1-2	Time of day-hours
3-4	Time of day-minutes
5	Duration-hours
6-7	Duration-minutes
8	Duration-tenths of minutes
9	Condition code
10-17	Space
18-32	Dialed number
33-42	Calling number
43-48	Space
49-55	Terminating SIP Entity
56-57	Space
58-64	Originating SIP Entity
65-73	Space
74	Feature flag
75-92	Space
93	BCC
94	MA-UUI
95	Resource flag
96-104	Space
105-106	Bandwidth
107-108	Space
109-144	Av-GSID
145-170	Space (Reserved for Future Use)
171-185	Calling Party IP
186-211	Space (Reserved for Future Use)
212-226	Called Party IP
227	Carriage return
228	Line feed

Field Descriptions: In the order of their appearance in the standard flat file CDR format.

Time of day (4 digits): The time (hh:mm) the call ended.

Duration (4 digits): The duration of the call in hours (0 to 9), minutes (00 to 59), and tenth of minutes (0 to 9). Session Manager rounds the duration of the call in 6- seconds increments. For example, the system records the duration of a 5-second call as 0000.

The duration field reports a 9999 if the call was in progress when a time change was made on the system and the call duration can no longer be determined. The duration field reports a 9599 for calls greater than 10 hours. See condition code 4 below.

Condition Code (1 alphanumeric character): Indicates the type of call that the call record describes, as listed below.

Table 6 - Condition Codes

Code	Description
4	Identifies a call of extremely long duration (a call that lasts for 10 or more hours). When a call exceeds 10 hours, SM creates a call record with this condition code and duration of 9 hours, 59 minutes, and 1-9 tenths of a minute. SM creates a similar call record with this condition code after each succeeding 10-hour period. When the call terminates, SM creates a final call record with the appropriate condition code identifying the type of call. This condition code does not apply when the output format is enhanced XML. A single call record for the complete call duration is created for the enhanced XML format regardless of the call length.
9	Identifies a tandem call. This is the condition code for calls that complete normally through SM. This condition code is also used for an incoming call.
A	Identifies an outgoing call from a SIP phone.

Note 1: The condition code “A” was added during SM Release 5.2 when the capability to handle “URE to SRE/non-URE” and “SRE to URE” calls were added to SM.

Note 2: Avaya recommends turning off CDR for the IMS trunks on the “change signaling group” page of the **CM SAT** screen when setting up the trunks in CM that will talk to SM. Otherwise, the **CM CDR** captures 2 or more call records for each call because of Application Sequencing in SM. These extra trunk records make processing the CM CDR records extremely difficult. There is no automated method to easily merge CM and SM CDR records.

Dialed Number (15 digits): The fifteen most significant digits of the numeric user portion of the request URI.

Calling Number (10 digits): The ten right-most digits of the numeric user portion identifying the calling number, taken either from the P-Asserted-Identity:header, Contact:header, or From:header.

Terminating SIP Entity (7 alphanumeric characters): The last seven characters of the outbound SIP Entity administered on the SM server.

Originating SIP Entity (7 alphanumeric characters): The last seven characters of the inbound SIP Entity.

Feature Flag (1 numeric digit): Set to “4” in the first SM release and identifies a voice call with network answer supervision.

Bearer Capability Class (1 alphanumeric character): Set to “M” in the first SM release, indicating a multimedia call.

MA-UUI (1 numeric digit): Set to “0” in the first SM release and indicates no message-associated user-to-user signaling.

Resource Flag (1 numeric digit): Set to “0” in the first SM release and indicate a circuit-switched call with no conversion device used. Beginning in SM 6.1, the Resource Flag is set to “4” whenever a call has had a video session during its lifetime.

Bandwidth (2 digits with a maximum value of ‘99’): Indicates the highest bandwidth used by the call, rounded to the nearest multiple of 64k (64,000), for example:

- 0 means < 32Kbps
- 1 means 64Kbps (± 32Kbps)
- 2 means 128Kbps (± 32Kbps)
- 16 means 1024Kbps (± 32Kbps) – this covers 1Mbps (1,000,000bps)
- 31 means 1984Kbps (± 32Kbps) – this covers 2Mbps (2,000,000bps)
- 32 means 2048Kbps (± 32 Kbps)
- 94 means 6016Kbps (± 32 Kbps) – this covers 6Mbps (6,000,000bps)
- 96 means 6144Kbps (± 32 Kbps)
- 99 means 6304Kbps or greater

Single-digit values are recorded with a leading blank (‘ 3’ and not ‘03’).

For an extremely long duration call (see condition code 4 in table 6) that results in multiple CDR records, the bandwidth field is correct in the final CDR record generated for the call.

Av-GSID (Avaya-Global-Session-ID**):** Upon receiving an initial SIP request (out of dialog), SM creates and stores a globally unique identifier (Av-GSID token). SM inserts the identifier in the Av-Global-Session-ID header in the request. If the incoming request already contains an Av-Global-Session-ID header, SM overwrites the header. An exception to the overwriting rule is when the request arrives from another SM, a sequenced application, or a trusted SIP entity. The Av-Global-Session-ID is present in the initial request that returns from the sequenced application.

The Av-global identifier (GSID token) in the Avaya-Global-Session-ID header is a 128-bit unique integer in time and space. SM follows one of the RFC 4122 (Universally Unique Identifier) recommended algorithms to generate the GSID token.

The GSID is a 36-character sequence comprised of digits 0-9, a-f, A-F,”-“.

Calling Party IP and Called Party IP: These fields each have a default width of 15 characters which enables them to capture today’s 15-character IPv4 addresses.

Any IP address populated into these fields is right justified with any “extra” character locations on the left of the address. The “extra” character locations is populated with “space” characters with an ASCII value of 32 decimal or 0x20 hex.

The format for IPv4 addresses is aaa.bbb.ccc.ddd where, “aaa”, “bbb”, ccc”, and “ddd” have 1-3 character decimal numbers whose values range between 0 and 255. Each of these three-digit numbers is separated by a period character (ASCII 46 or 0x2E hex).

An example of a typical IPv4 address in a CDR record is:	135.192.36.4
The minimal length address is 7 characters:	1.2.3.4
The maximum length address is 15 characters:	135.192.211.178

If the customer reduces the field size and the full IP address does not fit, it is truncated from the left for the low-order parts of the IP address to appear still.

Note that IPV6 is not supported as of SM 6.3.4.

Space (0x20), Carriage Return (0x0d), Line Feed (0x0a): the corresponding ASCII characters.

Table 7 - Enhanced Flat File Format in SM 6.3.4

Position	Description
1-2	Time of day-hours
3-4	Time of day-minutes
5	Duration-hours
6-7	Duration-minutes
8-9	Duration-seconds
10-12	Duration-milliseconds
13	Condition code
14	Space
15-29	Dialed number
30	Space
31-45	Calling number
46	Space
47-78	Terminating SIP Entity
79	Space
80-111	Originating SIP Entity
112	Space
113	Feature flag
114	Space
115	BCC
116	MA-UUI
117	Resource flag
118	Space
119-120	Bandwidth
121	Space
122-157	Av-GSID
158	Space
159-173	Calling Party IP
174	Space
175-189	Called Party IP
190	Space
191	Call usage-voice
192	Call usage-video
193	Call usage-fax
194	Call usage-text
195	Call usage-other
196	Space
197-199	Codec
200	Space
201-232	Calling party tenant id
233-264	Called party tenant id
265	Carriage return
266	Line feed

Field Descriptions: In the order of their appearance in the enhanced flat file CDR format.

Time of day: Same as a standard flat file format.

Duration (8 digits): The duration of the call, which the system records in hours (0 to 9), minutes (00 to 59), seconds (00 to 59), and milliseconds (0 to 999).

Condition Code (1 alphanumeric character): Indicates the type of call that the call record describes. The enhanced flat file CDR format supports the condition codes listed below in addition to those listed under the standard flat file format.

Table 8 - Condition Codes

Code	Description
6	Rejected – System resources (e.g., bandwidth) are unavailable
G	Incomplete call – Caller hung up while called station was ringing
I	Incomplete call – Called station was busy
N	Incomplete call – Call did not complete for some other reason. For example, called user was not registered.

Dialed Number: Same as a standard flat file format.

Calling Number: Same as a standard flat file format except that the field is expanded to 15 digits.

Terminating SIP Entity: Same as a standard flat file format except that the field is expanded to 32 characters.

Originating SIP Entity: Same as a standard flat file format except that the field is expanded to 32 characters.

Feature Flag: Same as a standard flat file format.

Bearer Capability Class: Same as a standard flat file format.

MA-UUI: Same as a standard flat file format.

Resource Flag: Same as a standard flat file format.

Bandwidth: Same as a standard flat file format.

Av-GSID: Same as a standard flat file format.

Calling Party IP and Called Party IP: Same as a standard flat file format.

Call Usage – voice (1 character): ‘Y’ if the call was a voice call at any time, otherwise, ‘N’.

Call Usage – video (1 character): ‘Y’ if the call was a video call at any time, otherwise, ‘N’.

Call Usage – fax (1 character): ‘Y’ if the call was a fax call at any time, otherwise, ‘N’.

Call Usage – text (1 character): ‘Y’ if call was a text call at any time, otherwise, ‘N’.

Call Usage – other (1 character): 'Y' if the type of call was something other than the above choices, otherwise, 'N'.

For an extremely long duration call (see condition code 4 in table 6) that results in multiple CDR records, the call usage fields is correct in the final CDR record generated for the call.

Codec (3 digits): The number of the codec used on the call. The codec number is extracted from the Session Description Protocol message body. For more information on the values of some commonly used codecs, see http://tools.ietf.org/html/rfc3551#page-32.

For an extremely long duration call (see condition code 4 in table 6) that results in multiple CDR records, the codec field is correct in the final CDR record generated for the call.

Calling and called tenant id (32 characters each): The administered tenant ids for the calling and called parties. These fields are blank if the respective party is not an administered SIP station user on Session Manager.

2.3 CDR XML FORMAT

CDR output is available in XML format starting with SM 6.3.4. The contents are functionally equivalent to the enhanced flat file format. The call time is reported as an XML 'dateTime' data type. The call duration is an XML 'duration' data type. For more information on the definitions of these data types, see http://www.w3.org/TR/xmlschema- 2/#built-in-primitive-datatypes. Because the call time element in the XML output gives the date and time of the call, there are no separate date records in the output.

The following is an example of XML schema till SM 7.1.2:

```
<?xml  version="1.0"  encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="calls">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="call" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="call_time" type="xs:dateTime"/>
                            <xs:element  name="duration"  type="xs:duration"/>
                            <xs:element name="condition_code">
                                <xs:simpleType>
                                    <xs:restriction  base="xs:string">
                                        <xs:pattern value="[469AGIN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="parties">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element  name="dialed_number"  type="xs:string"/>
                                        <xs:element  name="calling_number"  type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
```

```
<xs:element name="sip_entities">
      <xs:complexType>
           <xs:sequence>
                <xs:element  name="terminating"  type="xs:string"/>
                <xs:element  name="originating"  type="xs:string"/>
           </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="feature_flag">
     <xs:simpleType>

           <xs:restriction  base="xs:string">
                <xs:pattern value="[4]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="bcc">
     <xs:simpleType>
           <xs:restriction  base="xs:string">
                <xs:pattern value="[M]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="ma_uui">
     <xs:simpleType>
           <xs:restriction  base="xs:string">
                <xs:pattern value="[0]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="resource_flag">
     <xs:simpleType>
          <xs:restriction  base="xs:string">
                <xs:pattern value="[04]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="bandwidth">
     <xs:simpleType>
           <xs:restriction  base="xs:integer">
                 <xs:maxInclusive  value="99"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element  name="av_gsid"  type="xs:string"/>
<xs:element name="ip_addresses">
      <xs:complexType>
           <xs:sequence>
                <xs:element name="calling" type="xs:string"/>
                <xs:element name="called" type="xs:string"/>
           </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="usage">
```

```
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="voice">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[YN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="video">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[YN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="fax">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[YN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="text">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[YN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="other">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[YN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="codec" type="xs:string"/>
                <xs:element name="tenant_ids">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="calling" type="xs:string"/>
                            <xs:element name="called" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

```
    </xs:element>
</xs:schema>
```

The following is an example of an XML CDR output file containing a single call record till SM 7.1.2:

```
<?xml  version="1.0"  encoding="UTF-8"?>
<calls    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <call>
                <call_time>2002-05-30T09:30:10Z</call_time>
                <duration>PT3H27M1.246S</duration>
                <condition_code>4</condition_code>
                <parties>
                        <dialed_number>3035389999</dialed_number>
                        <calling_number>9702251234</calling_number>
                </parties>
                <sip_entities>
                        <terminating>cm4</terminating>
                        <originating>cm7</originating>
                </sip_entities>
                <feature_flag>4</feature_flag>
                <bcc>M</bcc>
                <ma_uui>0</ma_uui>
                <resource_flag>0</resource_flag>
                <bandwidth>31</bandwidth>
                <av_gsid>f652e050-a876-11e2-b585-00145e3e9b22</av_gsid>
                <ip_addresses>
        </call>         <calling>10.42.35.111</calling>
                        <called>123.43.1.76</called>
                </ip_addresses>
                <usage>
                        <voice>Y</voice>
                        <video>N</video>
                        <fax>N</fax>
                        <text>N</text>
                        <other>N</other>
                </usage>
                <codec>1</codec>
                <tenant_ids>
                        <calling>Joe's Bar and Grill</calling>
                        <called>One Hour Dry Cleaning</called>
                </tenant_ids>

</calls>
```

From 7.1.3 release, the CDR XML format includes two new fields: ucid and ucid2.

UCID (ucid, ucid2): UCID is an older Avaya approach for tagging calls across products. When inbound calls from PSTN (the key CC use case) fail before reaching CM, it becomes hard to troubleshoot the failure reason. Even though ASBCE generated and inserted User-to-User information in such call on the inbound leg and contain the UCID value, SM did not capture the User-to-User information in the SM-generated CDR. To address the above issue, SM is enhanced to capture the UCID values from the User-to-User header in the INVITE request and store it in the CDR.

Overview:
Based on administration, CM or SBC adds the User-to-User header in the INVITE request. When creating CDR, SM must take this header into account and populate the decoded value in CDR. The following is an example of a User-to-User header and its relevant UCID field in the CDR:

User-to-User: 00FA08000105335A7BFB5DFB08000105335A7BFB5E;encoding=hex
CDR element: <ucid>000105335A7BFB5D</ucid>
<ucid2>000105335A7BFB5E</ucid2>

There is a chance that the User-to-User header can contain applications other than ucid and ucid2. SM ignores those applications and records only ucid and ucid2. Applications are decided based on hex values present in the header. As shown in the example above, FA represents ucid, and FB represents ucid2. If the User-to-User header is not present in the call, then ucid and ucid2 elements will be added in CDR with no value. If the User-to-User header is present with only ucid, then <ucid> is added with a subsequent value, and <ucid2> is kept blank in CDR.

Length of ucid field: 16 characters
Length of ucid2 field: 16 characters

The following is an example of the XML schema since SM 7.1.3:

```
<?xml  version="1.0"  encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="calls">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="call" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="call_time" type="xs:dateTime"/>
                            <xs:element  name="duration"  type="xs:duration"/>
                            <xs:element name="condition_code">
                                <xs:simpleType>
                                    <xs:restriction  base="xs:string">
                                        <xs:pattern value="[469AGIN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="parties">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element  name="dialed_number"  type="xs:string"/>
                                        <xs:element  name="calling_number"  type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
```

```
</xs:element>
<xs:element name="sip_entities">
      <xs:complexType>
           <xs:sequence>
                <xs:element  name="terminating"  type="xs:string"/>
                <xs:element  name="originating"  type="xs:string"/>
           </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="feature_flag">
     <xs:simpleType>

           <xs:restriction  base="xs:string">
                <xs:pattern value="[4]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="bcc">
     <xs:simpleType>
           <xs:restriction  base="xs:string">
                <xs:pattern value="[M]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="ma_uui">
     <xs:simpleType>
           <xs:restriction  base="xs:string">
                <xs:pattern value="[0]"/>
           </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="resource_flag">
     <xs:simpleType>
          <xs:restriction  base="xs:string">
               <xs:pattern value="[04]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element name="bandwidth">
     <xs:simpleType>
           <xs:restriction  base="xs:integer">
                <xs:maxInclusive  value="99"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
<xs:element  name="av_gsid"  type="xs:string"/>
<xs:element name="ucid" type="xs:string"/>
<xs:element name="ucid2" type="xs:string"/>
<xs:element name="ip_addresses">
      <xs:complexType>
           <xs:sequence>
                <xs:element name="calling" type="xs:string"/>
                <xs:element name="called" type="xs:string"/>
           </xs:sequence>
      </xs:complexType>
```

```
                    </xs:element>
                    <xs:element name="usage">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="voice">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[YN]"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="video">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[YN]"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="fax">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[YN]"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="text">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[YN]"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="other">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:pattern value="[YN]"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="codec" type="xs:string"/>
                    <xs:element name="tenant_ids">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="calling" type="xs:string"/>
                                <xs:element name="called" type="xs:string"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
```

```
            </xs:complexType>
        </xs:element>
    </xs:schema>
```

The following is an example of an XML CDR output file containing a single call record since SM 7.1.3:

```
<?xml version="1.0" encoding="UTF-8"?>
<calls
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <call>
                <call_time>2002-05-30T09:30:10Z</call_time>
                <duration>PT3H27M1.246S</duration>
                <condition_code>4</condition_code>
                <parties>
                        <dialed_number>3035389999</dialed_number>
                        <calling_number>9702251234</calling_number>
                </parties>
                <sip_entities>
                        <terminating>cm4</terminating>
                        <originating>cm7</originating>
                </sip_entities>
                <feature_flag>4</feature_flag>
                <bcc>M</bcc>
                <ma_uui>0</ma_uui>
                <resource_flag>0</resource_flag>
                <bandwidth>31</bandwidth>
                <av_gsid>f652e050-a876-11e2-b585-00145e3e9b22</av_gsid>
                <ucid>000100205A44B42D</ucid>
                <ucid2>000100205A44B42C</ucid2>
                <ip_addresses>
                        <calling>10.42.35.111</calling>
                        <called>123.43.1.76</called>
                </ip_addresses>
                <usage>
                        <voice>Y</voice>
                        <video>N</video>
                        <fax>N</fax>
                        <text>N</text>
                        <other>N</other>
                </usage>
                <codec>1</codec>
                <tenant_ids>
                        <calling>Joe's Bar and Grill</calling>
                        <called>One Hour Dry Cleaning</called>
                </tenant_ids>
        </call>
</calls>
```

From 10.2 release, the CDR XML format includes two new fields: verstat and attestation_info.

STIR/SHAKEN (verstat, attestation_info): STIR/SHAKEN is an industry-standard caller ID authentication technology and is comprised of a set of technical standards and protocols that allow for the verification of caller ID information for calls carried over IP networks. Session Manager accepts the STIR/SHAKEN parameters in various service provider formats over SIP trunks and can normalize the

information into a standard Aura format by applying the StirShakenAdapter. The normalized format of parameters contained in the inbound INVITE request are then stored in the CDR.

Note: The STIR/SHAKEN parameters must be in normalized format in order to be captured properly in the CDR. Apply the StirShakenAdapter on service provider trunks providing STIR/SHAKEN information to ensure that the parameters appear in the CDR.

Overview:
Based on administration, Session Manager can accept and normalize the STIR/SHAKEN information contained in the INVITE request. In the normalized form, the *verstat* parameter is represented as a username parameter in the P-Asserted-Identify or From header URI and the *attestation*-info parameter is represented within the Attestation-Info header. The following is an example of a P-Asserted-Identity header with verstat parameter and Attestation-info header with attestation_info (also referred to as "attestation level"):

P-Asserted-Identify: "Caller Name" <+19702251234;verstat=TN-Validation-Passed@example.com>
Attestation-Info: A

If the verstat or attestation_info elements are not present in the call, then the verstat and attestation_info elements will be added in CDR with no value.

The valid values of verstat are: TN-Validation-Passed, TN-Validation-Failed, and No-TN-Validation.

The valid values of attestation_info are: A, B, and C.

Note: Since the verstat parameter is typically present in the P-Asserted-Identity or From header URI that also contains the calling party number information, Session Manager strips the verstat parameter from the URI before storing the calling party number in the CDR. This prevents duplicating the verstat information in the CDR.

The following is an example of the XML schema since SM 10.2:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="calls">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="call" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="call_time" type="xs:dateTime"/>
                            <xs:element name="duration" type="xs:duration"/>
                            <xs:element name="condition_code">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:pattern value="[469AGIN]"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:element>
                            <xs:element name="parties">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="dialed_number" type="xs:string"/>
                                        <xs:element name="calling_number" type="xs:string"/>
```

```
            </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="sip_entities">
        <xs:complexType>
            <xs:sequence>
                <xs:element  name="terminating"  type="xs:string"/>
                <xs:element  name="originating"  type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="feature_flag">
        <xs:simpleType>

            <xs:restriction  base="xs:string">
                <xs:pattern value="[4]"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="bcc">
        <xs:simpleType>
            <xs:restriction  base="xs:string">
                <xs:pattern value="[M]"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="ma_uui">
        <xs:simpleType>
            <xs:restriction  base="xs:string">
                <xs:pattern value="[0]"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="resource_flag">
        <xs:simpleType>
            <xs:restriction  base="xs:string">
                <xs:pattern value="[04]"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="bandwidth">
        <xs:simpleType>
            <xs:restriction  base="xs:integer">
                <xs:maxInclusive  value="99"/>
            </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element  name="av_gsid"  type="xs:string"/>
<xs:element name="ucid" type="xs:string"/>
<xs:element name="ucid2" type="xs:string"/>
<xs:element name="ip_addresses">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="calling" type="xs:string"/>
                <xs:element name="called" type="xs:string"/>
```

```
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="usage">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="voice">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[YN]"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="video">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[YN]"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="fax">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[YN]"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="text">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[YN]"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="other">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="[YN]"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="codec" type="xs:string"/>
    <xs:element name="tenant_ids">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="calling" type="xs:string"/>
                <xs:element name="called" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="verstat" type="xs:string"/>
    <xs:element name="attestation_info" type="xs:string"/>
```

```
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

The following is an example of an XML CDR output file containing a single call record since SM 7.1.3:

```
<?xml version="1.0" encoding="UTF-8"?>
<calls
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <call>
        <call_time>2002-05-30T09:30:10Z</call_time>
        <duration>PT3H27M1.246S</duration>
        <condition_code>4</condition_code>
        <parties>
            <dialed_number>3035389999@example.com</dialed_number>
            <calling_number>9702251234@example.com</calling_number>
        </parties>
        <sip_entities>
            <terminating>cm4</terminating>
            <originating>cm7</originating>
        </sip_entities>
        <feature_flag>4</feature_flag>
        <bcc>M</bcc>
        <ma_uui>0</ma_uui>
        <resource_flag>0</resource_flag>
        <bandwidth>31</bandwidth>
        <av_gsid>f652e050-a876-11e2-b585-00145e3e9b22</av_gsid>
        <ucid>000100205A44B42D</ucid>
        <ucid2>000100205A44B42C</ucid2>
        <ip_addresses>
            <calling>10.42.35.111</calling>
            <called>123.43.1.76</called>
        </ip_addresses>
        <usage>
            <voice>Y</voice>
            <video>N</video>
            <fax>N</fax>
            <text>N</text>
            <other>N</other>
        </usage>
        <codec>1</codec>
        <tenant_ids>
            <calling>Joe's Bar and Grill</calling>
            <called>One Hour Dry Cleaning</called>
        </tenant_ids>
        <verstat>TN-Validation-Passed</verstat>
        <attestation_info>A</attestation_info>
    </call>
</calls>
```

2.4 IPv6 STANDARD FLAT FILE

This format was introduced in SM 7.1.2. All the fields are the same as the standard flat file except for CallingIp and CalledIp. The CallingIP and CalledIP field is extended to 39 to accommodate IPv6 address.

2.5 IPv6 ENHANCED FLAT FILE

This format was introduced in SM 7.1.2. All the fields are the same as the Enhanced flat file except for CallingIp and CalledIp. The CallingIP and CalledIP field is extended to 39 to accommodate IPv6 address.

3. MINIMUM REQUIREMENTS

The following minimum requirements must be met to support the Session Manager CDR functionality described in this Interface Document:

- The SM server must be running the first or a subsequent release of Session Manager.
- There must be IP connectivity between the CDR adjunct and the target server from time to time to enable the remote collection of the CDR data files.

Please note that this document contains no minimum hardware or software specifications describing the revision or performance levels required for the CDR adjunct system. The specification of the minimum requirements of the adjunct is the CDR adjunct vendor's responsibility.

4. SECURITY PROVISIONS

If firewalls are implemented anywhere between the CDR adjunct and the various Session Manager servers, it may be necessary to "punch" pinholes in those firewalls to enable communications between the CDR adjunct and the servers. Please work with the network administrators to implement these pinholes if needed.

5. CDR DATA FILE NAMING AND STRUCTURE

The CDR data files generated by the Session Manager CDR feature are stored in a special directory[1] on the server created to store this information. Anytime the CDR adjunct logs into the server, it is given direct access to this directory.

The CDR files stored in the mentioned special directory are those CDR data files that the server has completed and closed and are now ready for the CDR adjunct to collect, process, and subsequently delete from the server. Members of the CDR_User group are assigned the necessary rights to read and subsequently delete the CDR data files.

The file naming convention used for the CDR data files before SM 6.3.4 and in SM 6.3.4 (for standard or enhanced flat file output format) is shown below:

tttssss-ssss-YYMMDD-hh_mm

Where:

The file name is fixed at 25 alphanumeric characters, including dashes (-) and underscore (_).

"ttt" is populated with the characters "S00".

[1] The full path of this special directory is "/var/home/ftp/CDR". The login of CDR adjunct is limited, and it can only access the files in this directory.

"ssss-ssss" is an alphanumeric string of four characters, followed by a dash (-), and followed by an alphanumeric string of four characters, for a total of nine characters. This string uniquely identifies the Session Manager server through its IPv4 IP address, in hexadecimal.

"YY" is a two-digit number representing the year when the file was created.

"MM" is a two-digit number representing the month when the file was created.

"DD" is a two-digit number representing the day of the month when the file was created.

"hh" is a two-digit number representing the hour of the day when the file was created (24-hour clock server time).

"mm" is a two-digit number representing the number of minutes after the hour when the file was created.

For example, for a CDR file created on January 29, 2009, at 3:24 PM from an SM at IP address 142.9.147.59, the data file is named as follows:

S008e09-933b-090129-15_24

The file name format is slightly different if the enhanced XML output file format is selected in SM 6.3.4. The first 25 characters of the name are determined as shown above. An additional character sequence of the form '_n.xml' is appended to the file name. 'n' is a number that starts at 1 and increments if more than one file with the same timestamp is generated. The '.xml' is a suffix to indicate that the file contains xml data. According to the example above, if there were three XML files generated with the same timestamp, the file names would be as follows:

S008e09-933b-090129-15_24_1.xml
S008e09-933b-090129-15_24_2.xml
S008e09-933b-090129-15_24_3.xml

6. DATA TRANSPORT PROTOCOLS

The CDR adjunct remotely logs onto the Session Manager server and uses the SFTP[2] protocol to transfer the CDR data files from the SM server to the CDR adjunct.

Client versions of the SFTP protocol are available for Unix, Linux, Windows, and Macintosh-based computer platforms. Please see the footnote for a partial list of available implementations of the SFTP protocol.

[2] "SSH File Transfer Protocol" Internet-Draft is available at http://www.ietf.org/internet- drafts/draft-ietf-secsh-filexfer-12.txt. A partial list of available SFTP clients that could be considered for this application is available at http://en.wikipedia.org/wiki/List_of_SFTP_clients.

7. CDR DATA FILE DELETION PROVISIONS

There are two “normal” provisions for removing CDR data files from the server. These methods, listed in order of preference, are:

- After the CDR adjunct retrieves a data file from the server and verifies that it contains valid data, it is recommended that the adjunct delete it from the server using the provisions of SFTP.

- Anytime the SM server detects that the CDR data files stored on its local hard drive are older than 5 days, the server automatically removes the identified CDR data file and record this activity in a log file named *cleanup.log*, available in the same directory as the CDR data files.

With the above “normal” methods of removing the CDR data files, an onsite or remote switch administrator or technician with proper login permissions can delete unwanted CDR data files with the appropriate BASH commands. To perform this activity, the administrator or technician must have root privileges or be a member of the CDR_User group.

8. OPERATIONAL PROVISIONS, PROCEDURES, AND CONCERNS

This section provides an example of a typical CDR data file retrieval session and suggestions and cautions concerning the setup and operation of the Session Manager CDR functionality.

8.1 CDR DATA FILE RETRIEVAL EXAMPLE

The following figure provides a screen capture of a typical manual CDR data file retrieval session from a single server named “MyServer.MyDomain.com”. All user inputs are shown as **bold** and highlighted text. All other text is shown as normal text.

```
C:\WINNT>sftp CDR@MySessionManager.MyDomain.com
Connecting to MySessionManager.MyDomain.com...
This system is restricted solely to authorized users for legitimate
business purposes only. The actual or attempted unauthorized access,
use, or modification of this system is strictly prohibited.


Unauthorized users are subject to company disciplinary procedures and
or criminal and civil penalties under state, federal, or other applicable
domestic and foreign laws.


The use of this system may be monitored and recorded for administrative
and security reasons.  Anyone accessing this system expressly consents
to such monitoring and recording, and is advised that if it reveals
possible evidence of criminal activity, the evidence of such activity
may be provided to law enforcement officials.


All users must comply with all corporate instructions regarding the
protection of information assets.
Password: XXXXXX
sftp>
sftp> mget S00*
Fetching /S008709-9332-090416-08_26 to S008709-9332-090416-08_26
Fetching /S008709-9332-090416-08_36 to S008709-9332-090416-08_36
Fetching /S008709-9332-090416-08_46 to S008709-9332-090416-08_46
Fetching /S008709-9332-090416-08_56 to S008709-9332-090416-08_56
Fetching /S008709-9332-090416-09_06 to S008709-9332-090416-09_06
Fetching /S008709-9332-090416-09_16 to S008709-9332-090416-09_16
Fetching /S008709-9332-090416-09_26 to S008709-9332-090416-09_26
```

```
Fetching /S008709-9332-090416-09_36 to S008709-9332-090416-09_36
Fetching /S008709-9332-090416-21_41 to S008709-9332-090416-21_41
sftp>
sftp> lls -al
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-08_26
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-08_36
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-08_46
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-08_56
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-09_06
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-09_16
-rwxr-----   1 Test    445213 Apr 20 11:18 S008709-9332-090416-09_26
-rwxr-----   1 Test     85873 Apr 20 11:18 S008709-9332-090416-09_36
-rwxr-----   1 Test       331 Apr 20 11:18 S008709-9332-090416-21_41
sftp>
sftp>
sftp> rm S008709-9332-090416-08_26
Removing /S008709-9332-090416-08_26
sftp> rm S008709-9332-090416-08_36
Removing /S008709-9332-090416-08_36
sftp> rm S008709-9332-090416-08_46
Removing /S008709-9332-090416-08_46
sftp> rm S008709-9332-090416-08_56
Removing /S008709-9332-090416-08_56
sftp> rm S008709-9332-090416-09_06
Removing /S008709-9332-090416-08_06
sftp> rm S008709-9332-090416-09_16
Removing /S008709-9332-090416-08_16
sftp> rm S008709-9332-090416-09_26
Removing /S008709-9332-090416-09_26
sftp> rm S008709-9332-090416-09_36
Removing /S008709-9332-090416-09_36
sftp> rm S008709-9332-090416-21_41
Removing /S008709-9332-090416-21_41
sftp>
sftp> bye

C:\WINNT>
```

Figure 1 - Typical SFTP CDR Data File Retrieval Session

The scenario depicted above consists of the following steps:

1. The user does a “cd” to the directory where the CDR data files are to be transferred.

2. The user runs the following command to begin the session:
   “`sftp CDR@MySessionManager.MyDomain.com`”

   The command launches the SFTP program and passes the Session Manager server name (“MySessionManager.MyDomain.com”) and the user id (“CDR”).

3. The server then responds with its standard login message. At the end of the login message, the server prompts the user for the password. The user then responds with the password.

4. The user is automatically placed in the CDR data files directory on the SM server.

5. Next, the “`mget S00*`” command is executed to copy all files that exist on the server with file names beginning with “S00” to the local directory. The server responds by copying the files with file names beginning with “S00” to the local directory.

6. Next, the user lists the *local* directory by entering the "`lls –al`**"** or equivalent command to view which files were transferred.

7. Now the user removes each of the transferred files one at a time from the server. This is done by running the command "`rm <individual file name>`".

   NOTE: The files are removed individually to avoid deleting a new CDR data file added to the directory between the time the files are transferred and the time they are deleted. If a new file was added during this period and the "`rm *`" command was used, the new file would be deleted without getting transferred to the CDR adjunct. Any CDR records that were contained in the "new" file would be permanently lost.

Caution: The CDR adjunct must perform some type of file verification function to ensure that the adjunct has accurately received the file before deleting it from the server. For example, this check could be comparing the relative file size of the received file with the file size on the server to ensure they are the same size before deleting the file on the server.

The adjunct provider may elect to design the adjunct to retrieve a group of files on one polling pass and then process those files to ensure they are in the correct format and contain valid data. If no errors were detected, the adjunct would remove the previously retrieved files on the next polling pass. If errors were detected in a file, the adjunct could re-retrieve the errant file before removing it from the server.

8. Finally, the adjunct runs the command "`bye`" to log out from the server.

8.2 SWITCH INFORMATION NEEDED TO ADMINISTER THE CDR ADJUNCT

The following information is obtained from the Switch Administrator to configure the CDR adjunct and retrieve the SM CDR data files:

- The username and password administered on the SM which is used by the CDR adjunct.
- A list of the fully qualified domain names or IP addresses for all the SM servers from where the adjunct collects the CDR data files.

8.3 FRIENDLY REMINDERS AND SUGGESTIONS

The following are suggestions of things to be considered when designing and installing the CDR adjunct to support the Session Manager CDR functionality:

- The SM administrator must create a special login and password and provide it to the CDR adjunct administrator so the adjunct can retrieve the CDR data files. The administrator should verify that this login and password work on each SM server from where the adjunct will collect the CDR files.

- The CDR Adjunct must query every SM server (administered with the CDR feature) to ensure that all CDR data files are collected. These queries must be completed regularly and timely to ensure that the server does not delete CDR data files before they are collected.

- You can automate the CDR records gathering from the various Session Manager servers using Expect[3] or some other scripting language.

- Please see the current Session Manager customer documentation for instructions on how to administer the CDR feature on the server. These documents can be found on the support.avaya.com website.

8.4 PRECAUTIONARY INFORMATION

Warning: It is critical that all Session Manager servers administered to support CDR are periodically queried for available CDR data files. Whenever CDR files are identified during the query process, they need to be retrieved on time, so they are not lost.

8.5 LIMITATIONS TO SM CDR

- When SM CDR records are required for SIP calls routed by an SM to or from a Communication Manager, it is imperative that Origination and Termination application sequences are utilized and that those sequences are properly administered.

[3] Expect is a Tcl based tool for automating and scripting interactive applications such as SFTP, SCP, and telnet. The Expect home page is located at https://www.nist.gov/services-resources/software/expect. Sources for versions of Expect for Unix, Linux, and Windows and possibly other operating systems are identified on the NIST web site.