Configuration Note

AudioCodes Digital VoIP Media Gateways

CAS Protocol Tables

For Mediant™ Series Digital PSTN Gateways

Version 7.2

ac audiocodes

Table of Contents

This page has been intentionally left blank.

> ## Notice
>
> Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Before consulting this document, check the corresponding Release Notes regarding feature preconditions and/or specific support in this release. In cases where there are discrepancies between this document and the Release Notes, the information in the Release Notes supersedes that in this document. Updates to this document and other documents as well as software files can be downloaded by registered customers at http://www.audiocodes.com/downloads.
>
> This document is subject to change without notice.
>
> Date Published: November-15-2017

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our Web site at www.audiocodes.com/support.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

> **Note:** Throughout this manual, unless otherwise specified, the term *device* refers to AudioCodes SIP Enterprise products that support digital PSTN trunk interfaces.

Document Revision Record

LTRT	Description
28614	Initial release for Version 7.2.
28615	Assigning CAS table per channel through CLI and REST.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our Web site at http://online.audiocodes.com/doc-feedback.

1   Introduction

The CAS Protocol tables contain the Channel-associated signaling (CAS) protocol definitions that are used for CAS-terminated trunks. This document provides an overview of CAS protocol tables, how to create them, load them to the device, and assign them to the device's trunks and / or B-channels.

This page has been intentionally left blank.

2 Constructing CAS Tables

The CAS protocol table file is a text file containing the protocol's state machine that defines the entire protocol process. It is constructed of States, predefined Actions/Events, and predefined functions. With this file, you have full control over CAS protocol and can define or modify any CAS protocol by writing the protocol state machine in a text file according to a few AudioCodes-defined rules.

You can use the supplied files or construct your own files, as described below:

➢ **To construct a CAS protocol file:**

1. Learn the protocol text file rules from which the CAS state machine is built.

2. Refer to the supplied CAS files for an example.

3. Build the specific protocol/script text file (for example, xxx.txt) file and its related numerical value h file (for example, UserProt_defines_xxx.h). Note that the xxx.txt file must include the following 'C include', for example:

   ```
   #include 'UserProt_defines_xxx.h')
   ```

4. Compile the xxx.txt with the TrunkPack Downloadable Conversion utility (DConvert) to create the xxx.dat file (see Section 'Creating a Loadable CAS Protocol Table File' on page 25).

> **Note:** The files **xxx.txt**, **CASSetup.h**, **cpp.exe** and **UserProt_defines_xxx.h** must be located in the same folder.

2.1 CAS Protocol Table Structure

This chapter describes the structure of the CAS Protocol table.

2.1.1 Protocol Table Elements

The *CASSetup.h* file includes all the predefined definitions necessary to build a new protocol text file or to modify an existing one.

The CAS protocol table file (xxx.txt) is composed of the following elements:

■ INIT Variables

■ Actions

■ Functions

■ States

2.1.1.1 INIT Variables

INIT variables are numeric values defined by users in UserProt_defines_xxx.h. These values can be used in the file xxx.txt.

For example, INIT_RC_IDLE_CAS defines the ABCD bits expected to be received in IDLE state. INIT_DTMF_DIAL defines the On-time and Off-time for the DTMF digits generated towards the PSTN. Refer to the detailed list in UserProt_defines_xxx.h and in the sample protocol text file (supplied CAS files). Refer to the following ST_INIT detailed explanation.

2.1.1.2 Actions

Actions (i.e., protocol table events) are protocol table events activated either by the DSP (e.g., EV_CAS_01) or by users (e.g., EV_PLACE_CALL, EV_TIMER_EXPIRED1). The full list of available predefined events is located in the file CASSetup.h.

2.1.1.3 Functions

Functions define a certain procedure that can be activated in any state or in the transition from one state to another. The available functions include, for example, SET_TIMER (timer number, timeout in milliseconds), SEND_CAS (AB value, CD value). A full list of the possible predefined functions can be found in the file CASSetup.h.

2.1.1.4 States

Each Protocol Table consists of several states that it switches between during the call setup and tear-down process. Every state definition begins with the prefix 'ST_' followed by the state name and colon. The body of the state is composed of up to four unconditional performed functions and a list of actions that may trigger this state.

Below shows an example taken from an E&M wink start table protocol file:

Table 2-1: ST_DIAL: Table Elements

Action	Function	Parameter		Next State
		#1	#2	
FUNCTION0	SET_TIMER	2	Extra Delay Before Dial	DO

Action	Function	Parameter		Next State
EV_TIMER_EXPIRED2	SEND_DEST_NUM	ADDRESS	None	NO_STATE
EV_DIAL_ENDED	SET_TIMER	4	No Answer Time	ST_DIAL_ENDED

When the state machine reaches the dial state, it sets timer number 2 and then waits for one of two possible actions to be triggered: Either timer 2 expiration or end of dial event. When timer 2 expires, the protocol table executes function SEND_DEST_NUM and remains in the same state (NEXT_STATE=NO_STATE). When the dial event ends, the protocol table sets timer 4 and moves to ST_DIAL_ENDED written in the field NEXT_STATE.

Although you can define your own states, there are two states defined in the file *CASSetup.h* that must appear in every protocol table created:

■ **ST_INIT:** When channels initialization is selected, the table goes into 'Init' state. This state contains functions that initialize the following global parameters:

- **INIT_RC_IDLE_CAS:** Defines the ABCD bits expected to be received in the IDLE state in the specific protocol. The third parameter used to enable detection of 4 bits` CAS value (see below).

- **INIT_TX_IDLE_CAS:** Defines the ABCD bits transmitted in IDLE state in the specific protocol.

- **INIT_DIAL_PLAN:** A change regarding the issue of an incoming call dialed number.

■ **ST_IDLE:** When no active call is established or is in the process of being established, the table resides in Idle state, allowing it to start the process of incoming or outgoing calls. When the call is cleared, the state machine table returns to its Idle state.

Process the incoming call detection event by declaring end of digit reception in the following ways (both for ADDRESS/destination number and ANI/source number):

■ Receiving '#' digit (in MF or DTMF).

■ The number of digits collected reaches its maximum value as defined in DIAL_PLAN parameter #1 and #2 for destination and ANI numbers respectively.

■ A predefined time-out value defined in DIAL_PLAN parameter #3 elapses.

■ In MFC-R2 reception of signal I-15 (depending on the variant).

Parameter	Description
INIT_DTMF_DIAL	Defines the On-time and Off-time for the DTMF digits generated towards the PSTN.
INIT_COMMA_PAUSE_TIME	Defines the delay between each digit when a comma is used as part of the dialed number string (see acPSTNPlaceCall for details).
INIT_DTMF_DETECTION	Defines the minimum/maximum On-time for DTMF digit dialing detection.
INIT_PULSE_DIAL_TIME	Not supported by the current stack version. Defines the Break and Make time for pulse dialing.
INIT_PULSE_DIAL	Not supported by the current stack version. Defines the Break and Make ABCD bits for pulse dialing.
INIT_DEBOUNCE	Defines the interval time of CAS to be considered (a stable one).

Parameter	Description
INIT_COLLECT_ANI	Enables or Disables reception of ANI in a specific protocol.
INIT_DIGIT_TYPE	The #1 parameter defines the dialing method used (DTMF, MF). With MFC-R2 protocols, this parameter is not applicable (digits are assumed to be R2 digits).
	The #2 parameter enabled to usage of SS5 tones (not used).
	The #3 parameter used to enable digits detection at the OutGoing side of the call (which needed at some protocols.
INIT_NUM_OF_EVENT_IN_STATE	Inserted for detection on TOTAL_NUMBER_OF_EVENTS_IN_STATE (CASSetup.h).
INIT_INIT_GLOBAL_TIMERS	Initiates specific timers; it is used with Parameter#1 for metering pulse timer duration.
INIT_PULSE_DIAL_ADDITIONAL_ PARAMS	Not used.
INIT_RINGING_TO_ANALOGUE	When using analogue gateway option, it defines the CAS value of ringing (#1) CAS value of silence (#2) and CAS value of polarity relevsal(#3).
INIT_DIGIT_TYPE_1	Defines the signaling system used to send operator service.
INIT_REJECT_COLLECT	Defines the method for reject collect calls: *disabled*, *using Line signaling,* or *using register signaling*.
INIT_VERSION	Defines the version number. The version number is relevant to the release version number and is a text information string (not related to the utility compilation version number).
INIT_SIZE_OF_TABLE_PARAM	Users must insert the definition of TOTAL_NUMBER_OF_EVENTS_IN_STATE from CASSetup.h.

2.1.2 Reserved Words

For reserved words such as DO, NO_STATE, etc., see the detailed list in CASSetup.h.

2.1.3 State Line Structure

Each text line in the body of each state comprises 6 columns:

1. Action/event
2. Function
3. Parameter #1
4. Parameter #2
5. Additional parameters
6. Next state

2.1.4 Action / Event

Action / event is the name of the table's events that are the possible triggers for the entire protocol state machine. These can be selected from the list of events in file CASSetup.h (e.g., EV_DISCONNECT_INCOMING).

At the beginning of the state, there can be up to four unconditional actions / events called FUNCTION. These events are functions that are unconditionally performed when the table reaches the state. These actions are labeled FUNCTION0 to FUNCTION3.

The following subsections provide a list of available protocols table actions (events to the state machine).

2.1.4.1   User Command Oriented Action / Event

Table 2-2: User Command Orientated Action / Event

User Command Oriented Action/Event	Description
EV_PLACE_CALL	When acpstnplacecall() is used.
EV_ANSWER	When acpstnanswercall() is used.
EV_MAKE_DOUBLE_ANSWER_CAS	When the function acpstnanswercall is used and the INIT_REJECT_COLLECT parameter is set to Line Signaling.
EV_MAKE_DOUBLE_ANSWER_MF	When the function acpstnanswercall is used and the INIT_REJECT_COLLECT parameter is set to Register Signaling.
EV_DISCONNECT	When function acpstndisconnectcall() is used and the call is outgoing.
EV_DISCONNECT_INCOMING	When function acpstndisconnectcall() is used and the call is incoming.
EV_RELEASE_CALL	When acpstnreleasecall() is used.
EV_FORCED_RELEASE	When accasforcedrelease () is used.
EV_USER_BLOCK_COMND	When accasblockchannel() is used. This event is used to block or unblock the channel.
EV_MAKE_METERING_PULSE	When the function accasmeteringpulse is used, it triggers the start of the metering pulse while using function set_pulse_timer to start the timer to get the off event (see event ev_metering_timer_pulse_off).
EV_METERING_TIMER_PULSE_OFF	An event sent after the timer (invoked by function set_pulse_timer) expires. Refer to ev_make_metering_pulse.
EV_MAKE_FLASH_HOOK	When accasflashhook is used, a flash hook is triggered.

2.1.4.2   CAS Change Oriented Events

Table 2-3: CAS Change Orientated Events

Event	Description
EV_CAS_1_1	A new cas a, b bits received (a=1, b=1, was stable for the bouncing period).
EV_CAS_1_0	A new cas a, b bits received (a=1, b=0, was stable for the bouncing period).
EV_CAS_0_1	A new cas a, b bits received (a=0, b=1, was stable for the bouncing

Event	Description
	period).
EV_CAS_0_0	A new cas a, b bits received (a=0, b=0, was stable for the bouncing period).
EV_CAS_1_1_1_1	A new cas a, b bits received (a=1, b=1, c=1, d=1 was stable for the bouncing period). To receive such detection (that is different from EV_CAS_1_1) you must set YES at the #3 parameter of INIT_RC_IDLE_CAS.

2.1.4.3 Timer Oriented Events

Table 2-4: Time-Orientated Events

Event	Description
EV_TIMER_EXPIRED1	Timer 1 that was previously set by the table has expired.
EV_TIMER_EXPIRED2	Timer 2 that was previously set by the table has expired.
EV_TIMER_EXPIRED3	Timer 3 that was previously set by the table has expired.
EV_TIMER_EXPIRED4	Timer 4 that was previously set by the table has expired.
EV_TIMER_EXPIRED5	Timer 5 that was previously set by the table has expired.
EV_TIMER_EXPIRED6	Timer 6 that was previously set by the table has expired.
EV_TIMER_EXPIRED7	Timer 7 that was previously set by the table has expired.
EV_TIMER_EXPIRED8	Timer 8 that was previously set by the table has expired.

2.1.4.4 Counter Oriented Events

Table 2-5: Counter Orientated Events

Event	Description
EV_COUNTER1_EXPIRED	The value of counter 1 reached 0.
EV_COUNTER2_EXPIRED	The value of counter 2 reached 0.

2.1.4.5 IBS Oriented Events

Table 2-6: IBS Orientated Events

Event	Explanation
EV_RB_TONE_STARTED	Ringback tone as defined in the Call Progress Tone *ini* file (type and index) is detected.
EV_RB_TONE_STOPPED	Ringback tone as defined in the Call Progress Tone *ini* file (type and index) is stopped after it was previously detected.
EV_BUSY_TONE	Not used.

Event	Explanation
EV_BUSY_TONE_STOPPED	Not used.
EV_FAST_BUSY_TONE	Not used.
EV_FAST_BUSY_TONE_STOPPED	Not used.
EV_ANI_REQ_TONE_DETECTED	R1.5 ANI-request tone as defined in the Call Progress Tone *ini* file (type and index) is detected.
EV_R15_ANI_DETECTED	R1.5 ANI digit-string was detected.
EV_DIAL_TONE_DETECTED	Dial tone as defined in the Call Progress Tone *ini* file (type and index) is detected.
EV_DIAL_TONE_STOPPED	Dial tone as defined in the Call Progress Tone *ini* file (type and index) is stopped after it was previously detected.

2.1.4.6   DTMF/MF Oriented Events

Table 2-7: DTMF / MF Orientated Events

Event	Explanation
EV_MFRn_0	MF digit 0 is detected (only DTMF & MFr1).
EV_MFRn_1	MF digit 1 is detected.
EV_MFRn_2	MF digit 2 is detected.
EV_MFRn_3	MF digit 3 is detected.
EV_MFRn_4	MF digit 4 is detected.
EV_MFRn_5	MF digit 5 is detected.
EV_MFRn_6	MF digit 6 is detected.
EV_MFRn_7	MF digit 7 is detected.
EV_MFRn_8	MF digit 8 is detected.
EV_MFRn_9	MF digit 9 is detected.
EV_MFRn_10	MF digit 10 is detected.
EV_MFRn_11	MF digit 11 is detected.
EV_MFRn_12	MF digit 12 is detected.
EV_MFRn_13	MF digit 13 is detected.
EV_MFRn_14	MF digit 14 is detected.
EV_MFRn_15	MF digit 15 is detected.
EV_MFRn_1_STOPPED	MF digit 1 previously detected is now stopped.
EV_MFRn_2_ STOPPED	MF digit 2 previously detected is now stopped.
EV_MFRn_3_ STOPPED	MF digit 3 previously detected is now stopped.
EV_MFRn_4_ STOPPED	MF digit 4 previously detected is now stopped.
EV_MFRn_5_ STOPPED	MF digit 5 previously detected is now stopped.

Event	Explanation
EV_MFRn_6_ STOPPED	MF digit 6 previously detected is now stopped.
EV_MFRn_7_ STOPPED	MF digit 7 previously detected is now stopped.
EV_MFRn_8_ STOPPED	MF digit 8 previously detected is now stopped.
EV_MFRn_9_ STOPPED	MF digit 9 previously detected is now stopped.
EV_MFRn_10_ STOPPED	MF digit 10 previously detected is now stopped.
EV_MFRn_11_ STOPPED	MF digit 11 previously detected is now stopped.
EV_MFRn_12_ STOPPED	MF digit 12 previously detected is now stopped.
EV_MFRn_13_ STOPPED	MF digit 13 previously detected is now stopped.
EV_MFRn_14_ STOPPED	MF digit 14 previously detected is now stopped.
EV_MFRn_15_ STOPPED	MF digit 15, previously detected is now stopped.
EV_END_OF_MF_DIGIT	When DialMF() is used and no more dialed number digits are available (they already were sent). For example, the far side requests the next ANI digit but all digits already have been sent. This event usually appears in MFC-R2 tables.
EV_FIRST_DIGIT	The first digit of the DNI / ANI number is detected.
EV_DIGIT_IN	An incoming digit (MFR1 or DTMF) is detected.
EV_WRONG_MF_LENGTH	An incoming digit was detected, but its duration (ON-TIME) is too long or too short.
EV_DIALED_NUM_DETECTED	The whole destination number is detected.
EV_ANI_NUM_DETECTED	The whole source number is detected.
EV_DIAL_ENDED	The dialing process finished and all digits dialed.
EV_NO_ANI	When DialMF() is used and no ANI is specified by the outgoing user in function acPSTNPlaceCall(). MFC

Note: MF digit includes MF R1, R2-FWD, or R2-BWD, according to the context, protocol type, and call direction.

The following actions / events cause the MFC-R2 table to send the correct MF tone to the backward direction:

Table 2-8: Actions / Events Causing MFC-R2 Table to Send Correct MF Tone to Backward Direction

Actions/Events	Explanation
EV_ACCEPT	When acCASAcceptCall is used (only in MFC-R2) with CALLED_IDLE as its reason parameter (for example, this sends MF backward B-6).
EV_ACCEPT_SPARE_MF1	When acCASAcceptCall is used with SPARE_MF1 as its reason parameter.
EV_ACCEPT_SPARE_MF9	When acCASAcceptCall is used with SPARE_MF9 as its reason parameter.
EV_ACCEPT_SPARE_MF10	When acCASAcceptCall is used with SPARE_MF10 as its reason parameter.
EV_ACCEPT_SPARE_MF11	When acCASAcceptCall is used with SPARE_MF11 as its reason parameter.
EV_ACCEPT_SPARE_MF12	When acCASAcceptCall is used with SPARE_MF12 as its reason parameter.
EV_ACCEPT_SPARE_MF13	When acCASAcceptCall is used with SPARE_MF13 as its reason parameter.
EV_ACCEPT_SPARE_MF14	When acCASAcceptCall is used with SPARE_MF14 as its reason parameter.
EV_ACCEPT_SPARE_MF15	When acCASAcceptCall is used with SPARE_MF 15 as its reason parameter.
EV_REJECT_BUSY	When acCASAcceptCall is used with CALLED_BUSY as its reason parameter.
EV_REJECT_CONGESTION	When acCASAcceptCall is used with CALLED_CONGESTION as its reason parameter.
EV_REJECT_UNALLOCATED	When acCASAcceptCall is used with CALLED_UNALLOCATED as its reason parameter.
EV_REJECT_SIT	When acCASAcceptCall is used with SIT as its reason parameter.
EV_REJECT_RESERVE1	When acCASAcceptCall is used with CALLED_RESERVE1 as its reason parameter.
EV_REJECT_RESERVE2	When acCASAcceptCall is used with CALLED_RESERVE2 as its reason parameter.

2.1.4.7   Operator Service Events (up to GR-506)

Table 2-9: Operator Service Events (Up to GR-506)

Event	Explanation
EV_SEND_LINE_OPERATOR_SERVICE1	Send operator service 1 (=Operator Released) using line signaling.

Event	Explanation
EV_SEND_LINE_OPERATOR_SERVICE2	Send operator service 2 (=Operator Attached) using line signaling.
EV_SEND_LINE_OPERATOR_SERVICE3	Send operator service 3 (=Coin Collect) using line signaling.
EV_SEND_LINE_OPERATOR_SERVICE4	Send operator service 4 (=Coin Return) using line signaling.
EV_SEND_LINE_OPERATOR_SERVICE5	Send operator service 5 (=Ring-back) using line signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE1	Send operator service 1 (=Operator Released) using register signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE2	Send operator service 2 (=Operator Attached) using register signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE3	Send operator service 3 (=Coin Collect) using register signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE4	Send operator service 4 (=Coin Return) using register signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE5	Send operator service 5 (=Ring-back) using register signaling.
EV_SEND_REGISTER_OPERATOR_SERVICE6	Send operator service 6 (=Coin Collect/Operator Released) using register signaling.

Note: The following actions/events are for internal use only:

- EV_INIT_CHANNEL
- EV_TO_USER
- EV_CLOSE_CHANNEL
- EV_OPEN_CHANNEL
- EV_FAIL_DIAL
- EV_FAIL_SEND_CAS
- EV_ALARM

2.1.5 Function

The function's column holds the name of the function to be activated when the action specified in the action / events field occurs. Select the functions from the list of eight functions defined in CasSetup.h (e.g., START_COLLECT). When NONE is specified in this column, no function is executed.

Note: Do not define the same timer number (by SET_TIMER) twice before the first one expires or is deleted.

2.1.6   Parameters

The following columns are used as the function's parameters:

■ **Parameter #1**

■ **Parameter #2**

The list of global parameters can be found in CasSetup.h. If a parameter is not essential, it can also be written as NONE.

> **Note:** To enable the dynamic format of the CAS file and reduce memory usage, you can only include the used parameters.

Table 2-10: Available User Functions and Corresponding Parameters

User Function	User Function Parameters and Descriptions
SET_TIMER	(Timer number, timeout). Sets the timers managed per B-channel. Their expiration triggers the state machine table. Each protocol table/state machine can use up to 8 timers per B-channel/call (timeout in msec) when the timers have 25 msec resolution.
SEND_CAS	(AB value, CD value). ABCD bits are sent as line signaling for the specific channel when the call is setup.
GENERATE_CAS_EV	Check the ABCD bits value, and send a proper event to the state machine.
SEND_EVENT	(Event type, cause). The specific event type is sent to the host/user and retrieved by applying the function acGetEvent().
SEND_DEST_NUM	En-bloc dialing: refers to the digits string located in function acPSTNPlaceCall. Three types are available: (1) DestPhoneNum (2) InterExchangePrefixNum (3) SourcePhoneNum.
DEL_TIMER	(Timer number). Deletes a specific timer or all the timers (0 represents all the timers) for the B-channel.
START_COLLECT	Initiates the collection of address information, i.e., the dialed (destination) number for incoming calls where appropriate, according to the protocol. In the time between START_COLLECT and STOP_COLLECT, no digit is reported to users (EV_DIGIT is blocked) and the destination number is reported in event EV_INCOMING_CALL_DETECTED.
STOP_COLLECT	Refer to START_COLLECT.
SET_COUNTER	(Counter number, counter value or NONE). Sets counters managed per B-channel. Their expiration triggers the state machine. The counter initialization value should be a non-negative number. To delete all timers, invoke this function with 0 in the counter number field.
DEC_COUNTER	(Counter number). Decreases the counter value by 1. When the counter value reaches 0, EV_COUNTERx_EXPIRES is sent to the table (where x represents the counter number).
RESTRICT_ANI	Indicate the incoming side to hide the ANI from the Far-end user.
SEND_MF	(MF type, MF digit or index or NONE, MF sending time). This function is used only with MFC-R2 protocols.

The Channel Parameter structure contains three parameters associated with sending digits:

Table 2-11: Parameters Associated with Sending Digits

Parameter	Description
AddressVector and ANIDigitVector	These parameters are initialized when function PlaceCall is used. When the code reaches the dialing section, it sends the MF digit according to the MF type specified in the MF type cell (the types are defined in file CASSetup.h): ▪ **ADDRESS:** Sends the digit from the address vector (destination number) according to the index requested. Refer to the Index definition. ▪ **ANI:** Sends the digit from the ANI vector (source number) according to the requested index. ▪ **SPECIFIC:** Sends the MF digit specified in the cell Parameter #2. ▪ **SOURCE_CATEGORY:** Sends the predefined source category MF digit. The source category digit is set as the parameter SourceNumberingType when function PlaceCall is used. The second and third parameters are ignored when this type is used. ▪ **TRANSFER_CAPABILITY:** Sends the predefined line category MF digit. The line category digit is set as the parameter TransferCapability when function PlaceCall is used. The second and third parameters are ignored when this type is used.
Index	Specifies the Offset of the next digit to be sent from the vector (ADDRESS or ANI types, described above): ▪ **Index 1:** Sends the next digit in the vector. ▪ **Index –n: S**ends the last n digit. Underflow can occur if n is greater than the number of digits sent so far. ▪ **Index 0:** Sends the last sent digit. ▪ **Index SEND_FIRST_DIGIT:** Starts sending the digits vector from the beginning (see CASSetup.h).
MF Send Time	This send time parameter specifies the maximum transmission time of the MF. ▪ **STOP_SEND_MF:** Stops sending the current MF. ▪ **SEND_PROG_TON:** Operation, Tone or NONE.

Two operations are available:

■ Sends the Call Progress Tone specified in the cell Parameter #2 (The second parameter can be taken from CASsetup.h)

■ Stops sending the last parameter

Parameter	Description
CHANGE_COLLECT_TYPE	(Collect Type). Used by the incoming user to indicate that waiting for receipt of the digit of the requested type. The type can be one of the following: ▪ **ADDRESS:** The user waits for receipt of address digits. ▪ **ANI:** The user waits for receipt of ANI digits. ▪ **SOURCE_CATEGORY:** The user waits for receipt of the source category. ▪ **TRANSFER_CAPABILITY:** The user waits for receipt of the source transfer capability (line category).

2.1.7   Next State

The Next State column contains the next state the table moves to after executing the function for that action/event line. When you select to stay in the same state, insert NO_STATE or use the current state.

Note the difference between NO_STATE and the current state name in this field. If you select to stay in the same current state, the unconditional actions (FUNCTION0) at the beginning of the state are performed. In contrast, NO_STATE skips these functions and waits for another action to arrive.

Reserved word 'DO' must be written in the next state field if the unconditional actions (FUNCTION0) at the beginning of the state are used.

2.1.8   Changing the Script File

■ CAS bouncing is filtered globally for each received CAS for each channel. Define the time for the filtering criteria in the protocol table file (see INIT_DEBOUNCE) and this exceeds the bouncing in the DSP detection of 30 msec.

■ ANI/CLI is enabled using parameter ST_INIT ANI with 'YES'. ANI/CLI is supported using EV_ANI_NUM_DETECTED as the table action for collecting the ANI number in an incoming call. For outgoing calls, the table's function SEND_DEST_NUM with ANI parameter l initiates ANI dialing. The ANI number is provided by you in the Source phone number parameter of acPSTNPlaceCall().

■ You can use ANSI C pre-compile flags such as #ifdef, #ifndef, #else and #endif in the CAS script file. For example, you can decide whether or not to play dial tone according to fulfillment of #ifdef statement. The definition itself must be in CASSetup.h.

2.1.8.1   MFC-R2 Protocol

■ Use the SEND_MF script function to generate the outgoing call destination number. In this case, the first parameter should be ADDRESS (or ANI for source phone number) and the second parameter –3 to 1 (+1), indicating which digit is sent out of the number that the string conveyed by you in acPSTNPlaceCall().

   • 1 (+1) implies sending of the next digit

   • 0 implies a repeat of the last digit

   • -1 implies the penultimate digit
     This parameter actually changes the pointer to the phone number string of digits. Thus, a one-to-one mapping with the MF backward signals of the R2 protocol exists.

■ Using parameter SEND_FIRST_DIGIT initiates resending the string from the beginning, (change the pointer back to first digit and then proceed as above). This parameter is defined in CASSetup.h.

■ When MFC-R2 protocol is used, the two detectors (opened by default) are the Call Progress Tones and MFC-R2 Forward MF. When you invoke an outgoing call via acPSTNPlaceCall(), MFC-R2 Forward MF detector is replaced with MFC-R2 Backward MF detector, since only two detectors per DSP channel are permitted to operate simultaneously.

■ The correct MF is automatically generated according to the call direction: Forward for outgoing calls and Backward for incoming calls.

- MFC-R2 protocol fault can cause a channel block. In this case, the script file (supplied) releases the call to enable the user to free the call resources and be notified as to being in blocking state.
- START_COLLECT and STOP_COLLECT must be used in the script file for MF collecting both in outgoing and incoming calls.

> **Warning:** If this script function isn't used, the script gets stuck and forward\backward MF are not detected.

- The Ringback Call Progress Tone is translated to a unique event acEV_PSTN_ALERTING, since the Ringback tone is actually used in all AudioCodes protocols' state machines. All other Call Progress Tones are conveyed via acEV_TONE_DETECTED and retrieved by the user according to their type and index (note that the Ringback tone should be defined in the Call Progress Tones table with the relevant type in order to get this event).
- When the tone detection event is received, users can perform any action. For example, if the event is received with BUSY tone indication, users can invoke acPSTNDisconnectCall() to end the call.
- The MFC-R2 destination number is collected using parameter EXPECTED_NUM_OF_DIGITS_MINUS_1 for SET_COUNTER that the user defines with UserProt_defines_R2_MF.h. The counter function is used to trigger the script file for the penultimate received. After receiving the last digit, the script file (acting as the outgoing register) initiates the A6/A3 FWD MF. Normally, variant supports end of digit information (MF15 or MF12) or silence at the end of the dialing (when MF15 is not used). A short pulse of MF3 (A3) is sent to indicate that the entire string of digits (according to Q442, 476) is received.
- Sending Group B digit by an incoming register requires invoking acCASAcceptCall() with a certain reason parameter. Six reason parameters are available:

Reason Parameter	Description
CALLED_IDLE	Subscriber's line is free. Continue the call sequence. Should usually be followed by accept or reject.
CALLED_BUSY	Subscriber line is busy. Perform disconnect procedures.
CALLED_CONGESTION	Congestion encountered. Perform disconnect procedures.
CALLED_UNALLOCATED	Dial number was not allocated. Perform disconnect procedures.
CALLED_RESERVE1	Reserved for additional group B (user additional requirements).
CALLED_RESERVE2	Reserved for additional group B (user additional requirements).

Each reason generates a specific action, defined by the user, who modifies the script file. The action is then used to generate/respond with a group B MF (free, busy, etc.).

■ **Transfer Capability:** This parameter under function acPSTNPlaceCAll() is used by the outgoing register to generate the service nature of the originating equipment. In most variants (countries), this is the same as the Calling Subscriber Categories, but in some countries it is different, such as in R2 China protocol where it is referred to as the KD (Group II) digit.

Note: This parameter only receives MF values from the enumerator acTISDNTransferCapability. Choose the MF digit according to the service type that should be sent.

■ **Source Category:** This parameter under function acPSTNPlaceCall() determines the calling subscriber category. For example, a subscriber with priority, a subscriber without priority, etc. The parameter is usually sent as part of the Group II forward digits (except for R2 China where it is sent as the KA digit using Group I forward digits).

Note: This parameter is applicable only to MFC-R2 protocol type.

This page has been intentionally left blank.
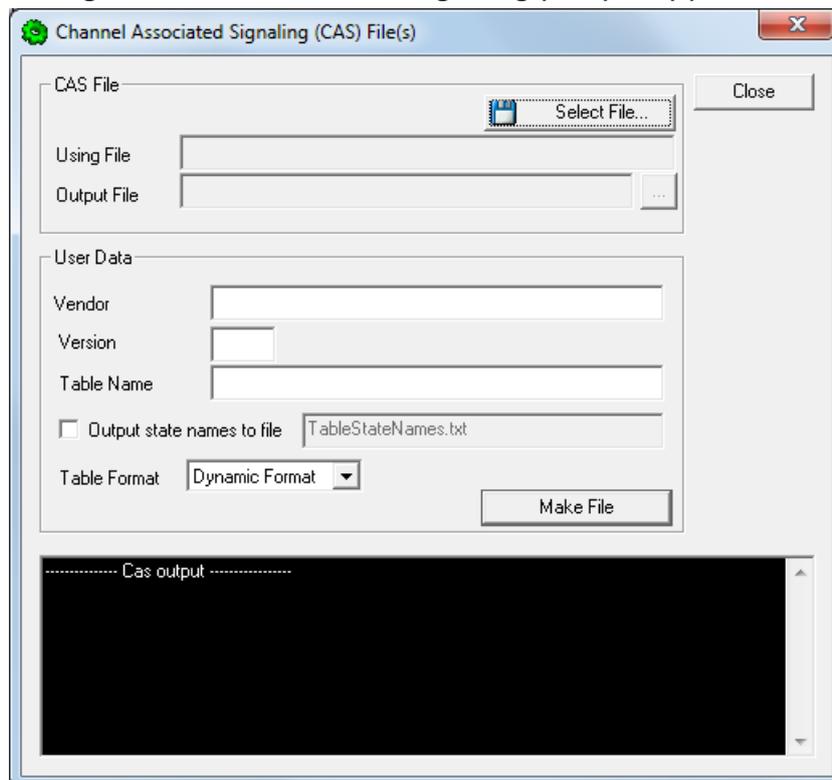
3   Creating Loadable CAS Table Files

The procedure below describes how to create a loadable CAS Protocol Table file using AudioCodes DConvert utility. For more information on this utility, refer to the document, *DConvert Utility User's Guide*.

➢ **To create a loadable CAS table file:**

1. Create the CAS protocol files (*xxx.txt* and *UserProt_defines_xxx.h*).

2. Copy the files generated in the previous step to the same directory in which DConvert is located. Ensure that the files *CASSetup.h* and *cpp.exe* are also located in the same directory.

3. Start DConvert.

4. Click **Process CAS Tables** button; the Channel Associated Signaling (CAS) File(s) screen opens:

Figure 3-1: Call Associated Signaling (CAS) File(s) Screen

5. Under the 'CAS File' group, click **Select File**, navigate to the folder in which the file is located, and then select the *txt* file you want converted; the 'Output File' field displays the file name and path, but with a *dat* extension. The table's name is also automatically designated.

6. Under the 'User Data' group, perform the following:

   a. In the 'Vendor' field, enter the vendor's name (maximum of 32 characters).

   b. In the 'Version' field, enter the version number. The value must be in the following format: [number] [single period '.'] [number] (e.g., 1.2, 23.4, 5.22)

7. In the 'Table Name' field, modify the name according to your requirements.

8. To create a file (for troubleshooting purposes) that contains the name of the States and their actual values, select the 'Output state names to file' check box; the default file name *TableStateNames.txt* appears in the adjacent field (you can modify the name of the file). The generated file is to be located in the same directory as DConvert.

9. From the 'Table Format' drop-down list, select the format you want to use:

   - Old Format: supported by all versions. Many CAS features are not supported in this format.

   - New Format: supported from 4.2 and later. From 5.2 and later a few new features are not supported by this format.

   - Dynamic Format: supported from 5.2 and later. Some 5.2 features are only supported by this format. The size of the file with dynamic format is significantly lower that other formats.

10. Click **Make File**; the *dat* file is generated and saved in the directory specified in the 'Output File' field. A message box informing you that the operation was successful indicates that the process is completed. In the pane at the bottom of the Call Assisted Signaling (CAS) Files(s) screen, the CAS output log box displays the log generated by the process. It can be copied as needed. The information in it isn't retained after the screen is closed.

4 Loading CAS Table Files to Device

The procedure below describes how to load the CAS Protocol table file(s) to the device through the Web interface. Up to eight files can be loaded to the device.

> **Note:** All CAS files loaded together must belong to the same trunk type (i.e., either E1 or T1).

> ➤ **To load CAS files to the device:**

1. Open the Auxiliary Files page (**Setup** menu > **Administration** tab > **Maintenance** folder > **Auxiliary Files**).

   Figure 4-1: Loading CAS File in CAS Table in Auxiliary Files Page

2. Click the **Browse** button corresponding to the 'CAS file' field, navigate to the folder in which the CAS file is located, and then click **Open**; the name and path of the file appear in the field next to the **Browse** button.

3. Click the **Load File** button corresponding to the 'CAS file' field.

4. Once the file load is complete, save the loaded file to the device's flash memory with a device reset.

This page has been intentionally left blank.

5 Assigning CAS Tables to Trunks and Channels

You can assign CAS tables to entire trunks or to specific B-channels. You can do this using different methods, where one requires you to first stop the trunk while the other method allows you to assign the CAS table on-the-fly.

5.1 Assigning CAS Tables by Stopping the Trunk

The following procedure describes how to assign a CAS table to an entire trunk and to assign different CAS tables to different channels of the trunk.

> **Note:** This method requires you to **stop** (de-activate) the trunks to which you are assigning the CAS tables. Therefore, this method is traffic effecting for all the B-channels of the trunk.

➢ **To assign CAS tables to trunks and B-channels:**

1. Open the Trunk Settings page (**Setup** menu > **Signaling & Media** tab > **Gateway** folder > **Trunks & Groups > Trunks**).
2. Stop the trunk by clicking the **Stop Trunk** button.
3. Under the 'CAS Configuration' group, select one of the following:
    *   To assign a CAS file to a trunk: Select the **CAS Table per Trunk** option, and then from the drop-down list, select the required file.
    *   To assign a CAS file to a B-channel of the trunk: Select the **CAS Table per Channel** option, and then from the drop-down list, select the required file. This parameter is a string, and can be set in one of the following formats:
        ♦   CAS table per channel: enter 31 indexes that point to the required CAS table for an E1 trunk (include dummy for B-channel 16), or 24 indexes for a T1 trunk. For example: "0,2,0,1,1,1,2,2,2,0,0,0,1,1,1,0,1,2,0,2,1,2,2,2"
            In this example, the first B-channel uses CAS table 0, the second B-channel uses CAS table 2, the third B-channel uses CAS table 0, and so on.
        ♦   CAS table per channel group: You can assign CAS tables to groups of channels (include 16) using the syntax x-y:z, where x-y is the channel range and z is the CAS table. For example: 1-10:2,11-20:7,21-31:2
            In this example, channels 1 through 10 use CAS table 2, and so on.

Figure 5-1: Selecting CAS Table in Trunk Settings Page

4. Click **Apply Trunk Settings**.
5. Save your settings to the device's flash memory with a device reset.

5.2   Assigning CAS Table per B-Channel On-the-Fly

You can assign a CAS table to a specific B-channel within a specific trunk, without stopping (de-activating) the trunk. In other words, channels that are not specified remain online and currently active calls on these channels continue uninterrupted.

> ⚠️ **Note:** When assigning a CAS table to the B-channel, all currently active calls on the specific channel are terminated.

You can assign a CAS table through the device's CLI or REST interface:

■ **Assigning a CAS table through CLI:**

1. Log in to the device's CLI in privileged-user mode.

2. Access the VoIP command set:

```
# configure voip
(config-voip)#
```

3. Access the required trunk:

```
(config-voip)# interface e1-t1 <Module>/<Trunk>
```

The following example accesses Trunk ID 2 on module (slot number) 1:

```
(config-voip)# interface e1-t1 1/2
(e1-t1 1/2)#
```

4. Type the following command:

```
(e1-t1 <Module>/<Trunk>)# set-cas-table-to-bchannel
<Channel Starting From 1> <CAS Index from 0>
```

The following example assigns CAS table index 2 to B-channel 5:

```
(e1-t1 1/2)# set-cas-table-to-bchannel 5 2
```

■ **Assigning a CAS table through the device's REST API:**

1. Prepare a CLI Script file with the following commands, replacing the variables (enclosed in angled brackets) with your required module (slot) number, trunk ID, B-channel, and CAS index:

```
configure voip
    interface e1-t1 <Module>/<Trunk>
          set-cas-table-to-bchannel <B-channel> <CAS Index>
          exit
    exit
exit
```

2. Connect to the device through REST.

3. Prepare your REST client to send the CLI Script file:

♦ Set the HTTP method to **PUT**.

♦ Set the method to **form-data**.

♦ Set the URL resource to:

http://<Device's IP Address>/api/v1/files/cliScript/incremental

♦ Load the CLI Script file that you prepared in Step 1 (above) to the REST client.

4. Load the file to the device by sending the configured PUT method.

6   Modifying Loaded CAS Tables

You can change the default settings of various timers and other basic parameters of each CAS table (state machine) file loaded to the device. This does not change the CAS state machine itself (no compilation is required).

> **Notes:**
>
> - Don't modify the default values unless you fully understand the implications of the changes and know the default values. Every change affects the configuration of the state machine parameters and the call process related to the trunk you are using with this state machine.
> - You can modify CAS state machine parameters only if the following conditions are met:
>   - √ Trunks are inactive (stopped), i.e., the 'Related Trunks' field displays the trunk number in green.
>   - √ State machine is not in use or is in reset, or when it is not related to any trunk. If it is related to a trunk, you must delete the trunk or de-activate (*Stop*) the trunk.
> - Field values displaying "-1" indicate CAS default values. In other words, CAS state machine values are used.
> - The modification of the CAS state machine occurs at the CAS application initialization only for non-default values (-1).

➢ **To modify the CAS state machine parameters:**

1. Open the CAS State Machine page (**Setup** menu > **Signaling & Media** tab > **Gateway** folder > **Trunks & Groups > CAS State Machines**).

Figure 6-1: CAS State Machine Page

CAS Table Name	Generate Digit On Time	Generate Inter Digit Time	DTMF Max Detection Time	DTMF Min Detection Time	Max Incoming Address Digits	Max Incoming ANI Digits
E_M_FGDWinkTable.dat	-1	-1	-1	-1	-1	-1
E_M_FGDWinkTable.dat	-1	-1	-1	-1	-1	-1
E_M_FGDWinkTable.dat	-1	-1	-1	-1	-1	-1

2. Ensure that the trunk is inactive. The trunk number displayed in the 'Related Trunks' field must be green. If it is red, indicating that the trunk is active, click the trunk number to open the Trunk Settings page, select the required Trunk number icon, and then click **Stop Trunk**.

3. In the CAS State Machine page, for each loaded CAS table, modify the required parameters according to the table below.

4. Once you have completed configuration, activate the trunk (if required):
   a. In the CAS State Machine page, click the trunk number displayed in the 'Related Trunks' field; the Trunk Settings page opens.
   b. In the Trunk Settings page, click **Apply Trunk Settings**.

5. Click **Submit**, and then reset the device.

Table 6-1: CAS State Machine Parameters Description

Parameter	Description
Generate Digit On Time **[CasStateMachineGenerateDigitOnTime]**	Generates digit on-time (in msec). The value must be a positive value. The default value is -1 (use value from CAS state machine).
Generate Inter Digit Time **[CasStateMachineGenerateInterDigitTime]**	Generates digit off-time (in msec). The value must be a positive value. The default value is -1 (use value from CAS state machine).
DTMF Max Detection Time **[CasStateMachineDTMFMaxOnDetectionTime]**	Detects digit maximum on time (according to DSP detection information event) in msec units. The value must be a positive value. The default value is -1 (use value from CAS state machine).
DTMF Min Detection Time **[CasStateMachineDTMFMinOnDetectionTime]**	Detects digit minimum on time (according to DSP detection information event) in msec units. The digit time length must be longer than this value to receive a detection. Any number may be used, but the value must be less than CasStateMachineDTMFMaxOnDetectionTime. The value must be a positive value. The default value is -1 (use value from CAS state machine).
MAX Incoming Address Digits **[CasStateMachineMaxNumOfIncomingAddressDigits]**	Defines the limitation for the maximum address digits that need to be collected. After reaching this number of digits, the collection of address digits is stopped. The value must be an integer. The default value is -1 (use value from CAS state machine).
MAX Incoming ANI Digits **[CasStateMachineMaxNumOfIncomingANIDigits]**	Defines the limitation for the maximum ANI digits that need to be collected. After reaching this number of digits, the collection of ANI digits is stopped. The value must be an integer. The default value is -1 (use value from CAS state machine).
Collet ANI **[CasStateMachineCollectANI]**	In some cases, when the state machine handles the ANI collection (not related to MFCR2), you can control the state machine to collect ANI or discard ANI. ▪ **[0]** No = Don't collect ANI. ▪ **[1]** Yes = Collect ANI. ▪ **[-1]** Default = Default value - use value from CAS state machine.
Digit Signaling System **[CasStateMachineDigitSignalingSystem]**	Defines which Signaling System to use in both directions (detection\generation). ▪ **[0]** DTMF = Uses DTMF signaling. ▪ **[1]** MF = Uses MF signaling (default). ▪ **[-1]** Default = Default value - use value from CAS state machine.

This page is intentionally left blank.

International Headquarters
1 Hayarden Street,
Airport City
Lod 7019900, Israel
Tel: +972-3-976-4000
Fax: +972-3-976-4040

AudioCodes Inc.
27 World's Fair Drive,
Somerset, NJ 08873
Tel: +1-732-469-0880
Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide
Website: https://www.audiocodes.com/

Document #: LTRT-28615