QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Linux based 3G Specification

Multimedia Mobile Phone API

Circuit Switched Communication Service

Document: CELF_MPP_CS_D_V2.2_20051003

Draft 2.2.1

NEC Corporation

Panasonic Mobile Communication Ltd.

Revision History

Revision	Comment	Reviewer	Editor	Date
2.2	Initial	F2F meeting	NEC/Panasonic	05/09/28
2.2.1	Editorial Changes		AK	05/10/03

Introduction

Circuit Switched Communication Service (CS Service) has the function of the call control, the call state management, the tone control and the log processing.

Circuit Switched Communication Service includes Voice communication service, Video communication service, and Unrestricted Digital data Communication service.

References

Normative

RFC 2119: “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

RFC 2234: “Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November 1997, URL:http://www.ietf.org/rfc/rfc2234.txt

Informative

<as needed>

0. General

0.1 Event Structure

0.1.1 Circuit switched status notification event structure

```
typedef struct{
 int category;       The value is VoiceNotify
 int subtype;        The value is VoiceNotify_ConnInfo
 int info;           Voice communication status
 int subinfo;        Linetype
         union {
         ...
         } data ;  Unused
 }_CELF_CS_EVENT;
```

0.1.2 Voice communication status (MpComStatus)

The mobile phone can handle maximum three calls. This is called the multiple calls.

In case that one call is AV call, the mobile phone handles this call only.

0.1.2.1 Condition: only one call

CELF_CS_COM_STATUS_WAIT:	Standby
CELF_CS_COM_STATUS_RCV:	Under incoming
CELF_CS_COM_STATUS_TRN:	Under outgoing
CELF_CS_COM_STATUS_DLV:	Under calling
CELF_CS_COM_STATUS_TLK:	Under conversation
CELF_CS_COM_STATUS_HLD:	Under response hold

(This status is (a) that incoming call was received, and (b) that this incoming call cannot transit to conversation status because of the mobile phone.)

CELF_CS_COM_STATUS_RLS:	Under release

0.1.2.2 Condition: two call

One call is in conversation, and another call is in some status.

CELF_CS_COM_STATUS_TLK_RCV:	Under conversation and incoming
CELF_CS_COM_STATUS_TLK_TRN:	Under conversation and outgoing
CELF_CS_COM_STATUS_TLK_DLV:	Under conversation and calling
CELF_CS_COM_STATUS_TLK_RSV:	Under conversation and hold
CELF_CS_COM_STATUS_TLK_RLS:	Under conversation and release

- three call   One call is in conversation, another call is in hold, and 3rd call is in incoming.

CELF_CS_COM_STATUS_TLK_RSV_RCV:	Under conversation, hold, and incoming

0.1.2.3 Condition: only one AV call

CELF_CS_COM_STATUS_RCV_AV:	Under incoming of an AV call
CELF_CS_COM_STATUS_TRN_AV:	Under outgoing of an AV call
CELF_CS_COM_STATUS_DLV_AV:	Under calling of an AV call
CELF_CS_COM_STATUS_TLK_AV:	Under conversation of an AV call
CELF_CS_COM_STATUS_HLD_AV:	Under response hold of an AV call
CELF_CS_COM_STATUS_RLS_AV:	Under release of an AV call

Other voice communication call is not defined. For example, the VCS is not defined

(a) that one call is in incoming and another call is in outgoing,

(b) that two call are both in conversation,

(c) that two call are in hold and other call is in conversation, and so on.

0.1.2.4 Line type

CELF_CS_LINE_WCDMA	WCDMA

0.1.3 Call duration notification event structure

```
typedef struct{
 int category;        The value is VoiceNotify
 int subtype;         The value is VoiceNotify_TelCallTime
 int info;            Call duration (seconds)
 int subinfo;         Unused
            union {
            ...
            } data ;  Unused
 }_CELF_CS_EVENT;
```

<Disconnection cause notification>

0.1.4 Disconnection cause notification event structure

```
typedef struct{
 int category;        The value is VoiceNotify
 int subtype;         The value is VoiceNotify_DiscCause
 int info;            Call reference
 int subinfo;         Unused
            union {
```

```
        _CELF_CS_DISC_CAUSE cme;   Disconnection cause information structure
        } data ;
 }_CELF_CS_EVENT;
```

0.1.5 Disconnection cause information structure

```
typedef struct {
 unsigned char e_code;      Result code flag
 unsigned char code;        Result code
 unsigned char e_cause1;    Error reason 1 flag
 unsigned char cause1;      Error reason 1 (ccpMtCause)
 unsigned char e_cause2;    Error reason 2 flag
 unsigned char cause2;      Error reason 2 (Cause)
} _CELF_CS_DISC_CAUSE ;
```

0.1.6 Forwarding result notification event structure

```
typedef struct{
 int category;      The value is VoiceNotify
 int subtype;       The value is VoiceNotify_FW_Result
 int info;          Call reference
 int subinfo;       Forwarding result
       union {
       _CELF_CS_FW_RESULT fw_result;   Forwarding result structure
       } data ;
}_CELF_CS_EVENT;
```

0.1.7 Forwarding result

CELF_CS_OK	Successful forwarding
CELF_CS_ERR	Forwarding failure

0.1.8 Forwarding result structure

```
typdef struct {
 int cause ;        forwarding result details
} _CELF_CS_FW_RESULT ;
```

0.1.9 Forwarding result details (*Set only at forwarding failure.)

CELF_CS_FW_ERROR_NO_JOIN	Service is not contracted.
CELF_CS_FW_ERROR_NO_SETDATA	The forwarded destination is not registered.
CELF_CS_FW_ERROR_ETC	Others

<Off-hook transmission timeout notification>

0.1.10 Off-hook transmission timeout event structure

```
typedef struct{
 int category;        The value is  VoiceNotify
 int subtype;         The value is  VoiceNotify_OffHk_Trn
 int info;            Call reference
 int subinfo;         Communication type
            union {
            ...
            } data ;
}_CELF_CS_EVENT;
```

0.1.11 Communication type (CELF_CS_BTYPE)

CELF_CS_BTYPE_CS_NONE	None (unfixed)
CELF_CS_BTYPE_CS_ANY	Not Specified
CELF_CS_BTYPE_CS_VOICE	Voice
CELF_CS_BTYPE_CS_UD32UD	32K communication
CELF_CS_BTYPE_CS_UD64UD	64K communication
CELF_CS_BTYPE_CS_AV32AV	32K communication
CELF_CS_BTYPE_CS_AV64AV	64K communication

0.1.12 Connection Destination Information

```
typedef struct {
 int CN_No;                    // Call reference
 int CN_status;
 int continue_flag;
 unsigned char Calling_Dail [CELF_CS_DIAL_MEX+1];
 unsigned char Called_Dail [CELF_CS_DAIL_MAX+1];
 unsigned char BTsound_inf;
 CELF_CS_BTYPE bc_type;
 unsigned char taf_address;
unsigned char Cause_of_NoCLI;
unsigned char num_presentation_indicatior;
unsigned char redirectnum [CELF_CS_DIAL_MAX+1];
unsigned char redirect_presentation_indicator;
unsigned char signal;
 T_CELF_CS_CME cause; // Disconnection cause information structure
} _CELF_CS_CONNECT_INF
```

0.1.13 Call Reference Status

CELF_CS_USED:	”CN_No” is valid.
CELF_CS_UNUSED:	“CN_No” is not valid.

When Call reference status is unused, there is no connection between this mobile phone and other party. In this case, all data is void.

0.1.14 Call Status

Call status for this mobile phone

CELF_CS_CHAN_KIND_NULL:	Vacant
CELF_CS_CHAN_KIND_OFF:	Off-hook
CELF_CS_CHAN_KIND_TRN:	Outgoing call
CELF_CS_CHAN_KIND_DLV:	Calling
CELF_CS_CHAN_KIND_RCV:	Incoming call
CELF_CS_CHAN_KIND_REQ_T:	Response (conversation)

(The status of responding mobile phone is conversation.)

CELF_CS_CHAN_KIND_ACT:	Under conversation
CELF_CS_CHAN_KIND_REQ_H:	Response (hold)

(The status of responding mobile phone is hold.)

CELF_CS_CHAN_KIND_HLD:	Hold response
CELF_CS_CHAN_KIND_RSV:	Under hold
CELF_CS_CHAN_KIND_REL:	Under release

0.1.15 Existence of continuation data

CELF_CS_ON:	valid below data
CELF_CS_OFF:	non valid below data

The below data, from "Calling_Dail" to "cause", are valid data if the call status is incoming or conversation and incoming call.

0.1.16 Dial Number

Dial number of other party

This data is valid when this mobile phone originates.

CELF_CS_DIAL_MAX is 45.

0.1.17 BT sound flag

Whether BT sounds in this phone, or not

CELF_CS_SOUND_BT_ON:	BT tone sounds.
CELF_CS_SOUND_BT_OFF:	BT tone is being stopped.

0.1.18 TAF address

Internal/External TAF type

32 to 63:	Internal TAF
64 to 79:	External TAF

0.1.19 Cause of NoCLI

The reason why the dial number of other party is not notified.

The dial number of other party is in "Calling dial" or "Called dial".

CELF_CS_NOCL_NOSRV:	Because that the service is not supported.
CELF_CS_NOCL_USER:	Because that the user rejects to display.
CELF_CS_NOCL_INTRACTSRV:	Because that the service conflicts.
CELF_CS_NOCL_PAYPHON:	Because that the origination is from a public phone.

This data is valid, when next data “num_presentation_indicator”, is that Display is impossible.

0.1.20 Dial number display identifier

Whether dial number of other party can be displayed, or not.

CELF_CS_PRSNT_IND_ALLOWED:	Displayable
CELF_CS_PRSNT_IND_RESTRICTED:	Impossible to display
CELF_CS_PRSNT_IND_NOT_AVAILABLE:	Displayable number does not exist.
CELF_CS_PRSNT_IND_RESERVE:	Reservation

0.1.21 Redirection number

Destination number of call transfer.

redirectnum [CELF_CS_DAIL_MAX+1]

CELF_CS_DIAL_MAX is 45.

0.1.22 Redirect number display identifier

Whether redirection number can be displayed, or not.

CELF_CS_PRSNT_IND_ALLOWED:	Displayable
CELF_CS_PRSNT_IND_RESTRICTED:	Display is impossible.
CELF_CS_PRSNT_IND_NOT_AVAILABLE:	Displayable number does not exist.
CELF_CS_PRSNT_IND_RESERVE:	Reservation

0.1.23 Signal information

The type of tone of this phone

CELF_CS_SIGNAL_DIAL_TONE_ON:	Dial tone on
CELF_CS_SIGNAL_RINGBACK_TONE_ON:	Ring back tone on
CELF_CS_SIGNAL_INTERCEPT_TONE_ON:	Intercept tone on
CELF_CS_SIGNAL_NW_CONGESTION_TONE_ON:	Network congestion tone on
CELF_CS_SIGNAL_BUSY_TONE_ON:	Busy tone on
CELF_CS_SIGNAL_CONFIRM_TONE_ON:	Confirm tone on
CELF_CS_SIGNAL_ANSWER_TONE_ON:	Answer tone on
CELF_CS_SIGNAL_CALLWAITING_TONE_ON:	Call waiting tone on
CELF_CS_SIGNAL_OFFHK_WARNING_TONE_ON:	Off-hook warning tone on
CELF_CS_SIGNAL_TONES_OFF:	Tones off
CELF_CS_SIGNAL_ALERTING_OFF:	Alerting off
CELF_CS_SIGNAL_UNSETTING:	Signal information is not set.

0.1.24 Connection Request (CELF_CON_REQ)

```
typedef struct {
  CELF_CS_BTYPE       type;
 unsigned char *      dial_buf;
 int                  dial_len;
 CELF_NOTICE          notice;
 unsigned char *      subaddr_buf;
 int                  subaddr_len;
} _CELF_CON_REQ
```

0.1.25 Originating Number notification (CELF_NOTICE)

Whether the originating dial number is notified or not.

CELF_CS_NOTICE_ON:	Notified
CELF_CS_NOTICE_OFF:	Not notified
CELF_CS_NOTICE_NOSET:	No setting

0.1.26 Channel Number Information

CELF_CS_CHANNUM is used to hold call reference information.

If a channel is not used, CELF_CS_CHAN_NOUSE is set as the call reference.

```
Typedef struct {
  int ChanNum_00          // Call reference information 00
 int ChanNum_01           // Call reference information 01
 int ChanNum_02           // Call reference information 02
} _CELF_CS_CHANNUM
```

0.1.27 DCF Event Structure

```
typedef struct{
int  category;      The value is VoiceNotify
int subtype;        Event type
int info;           Notification type
int subinfo;        Bearer type
union {
 ...
  <DCF message structure corresponding to report types >
 ...
 } data ;
} CELF_CS_EVENT;
```

0.1.28 DCF Event Type

VoiceNotify_DCF_Disp	Display-related message
VoiceNotify_DCF_History	History-related message
VoiceNotify_DCF_Tone1	Tone 1-related message
VoiceNotify_DCF_Tone2	Tone 2-related message
VoiceNotify_DCF_ETC	Other messages

0.1.29 CCP Notification type

CELF_CS_CCP_CALLING_START_REQ	Notification of starting display during CCP outgoing
CELF_CS_CCP_CALLED_START_IND	Notification of starting display during CCP incoming
CELF_CS_CCP_CALLING_ALERTING_IND	Notification of starting display during CCP calling
CELF_CS_CCP_CONNECT_START_RSP	Notification of starting display during CCP connection
CELF_CS_CCP_CONNECT_START_IND	Notification of starting display during CCP communication
CELF_CS_CCP_RELEASE_IND	Notification of ending CCP display
CELF_CS_CCP_DISCONNECT_REQ	Notification of starting CCP disconnection (on a mobile device) display
CELF_CS_CCP_DISCONNECT_START_IND	Notification of starting CCP disconnection (on a network) display
CELF_CS_CCP_CALLING_REJ_IND	Notification of rejecting CCP outgoing
CELF_CS_CCP_HOLD_CNF	Notification of CCP hold
CELF_CS_CCP_RETREIVE_CNF	Notification of releasing CCP hold
CELF_CS_CCP_CALLING_SETUP_REQ	Notification of registering CCP outgoing call history
CELF_CS_CCP_CALLED_REJ_REQ	Notification of registering CCP absence incoming call history
CELF_CS_CCP_CALLED_SETUP_RSP	Notification of registering CCP incoming call history
CELF_CS_CCP_RGT_START	Notification of CCP RGT start
CELF_CS_CCP_RGT_STOP	Notification of CCP RGT stop
CELF_CS_CCP_HRGT_START	Start notification of incoming of a CCP hold call
CELF_CS_CCP_HRGT_STOP	Stop notification of incoming of a CCP hold call
CELF_CS_CCP_DST_START	Notification of CCP DST start
CELF_CS_CCP_DST_STOP	Notification of CCP DST stop
CELF_CS_CCP_RBT_START	Notification of CCP RBT start
CELF_CS_CCP_RBT_STOP	Notification of CCP RBT stop
CELF_CS_CCP_BT_START	Notification of CCP BT start
CELF_CS_CCP_CWT_START	Notification of CCP CWT start
CELF_CS_CCP_CWT_STOP	Notification of CCP CWT stop
CELF_CS_CCP_REJECT_ASK	Inquiry report of rejecting a CCP CS incoming call

0.1.30 Line status change notification event structure

```
typedef struct{
int  category;       // The value is VoiceNotify
int subtype;         // The value is VoiceNotify_AreaInfo
int info;           // Line status
int subinfo;        // Line type
        union {
        ...
        } data ; // Unused
}CELF_CS_EVENT;
```

0.1.31 Line status

CELF_CS_LINE_STATUS_OUT:	Out-of-communication area
CELF_CS_LINE_STATUS_IN:	Within-communication area

0.1.32 Line type

CELF_CS_LINE_WCDMA	WCDMA

0.1.33 Restriction status change notification event structure

```
typedef struct{
int  category;      // The value is VoiceNotify
int subtype;        // The value is VoiceNotify_Restrict
int info;           // Notification type
int subinfo;        // Restriction status
        union {
        ...
        CELF_CS_RES_CHG_INF       res_chg_inf;      // Restriction display information structure
        } data ;
}CELF_CS_EVENT;
```

0.1.34 Notification type

CELF_CS_RSMP_REST_STA:	Restriction display start notification
CELF_CS_RSMP_REST_END:	Restriction display end notification

0.1.35 Restriction status

The 0th bit is used for PS restriction status, and the 1st bit is used for CS restriction status.

(Bit ON means "restricted." Bit OFF means "unrestricted.")

CELF_CS_BIT_RESTINF_CS:	CS restriction information

CELF_CS_BIT_RESTINF_PS: PS restriction information

The 2nd bit is used for PS emergency restriction status, and the 3rd bit is used for CS emergency restriction status.

CELF_CS_BIT_ECRESTINF_CS: Emergency CS restriction information

CELF_CS_BIT_ECRESTINF_PS: Emergency PS restriction information

0.1.36 Restriction display information structure

typedef struct {

unsigned charNcRestriction; Normal originating restriction

unsigned charServiceStatus; Service status

unsigned charEcRestriction; Emergency originating restriction

} _CELF_CS_RES_CHG_INF ;

0.1.37 Normal and emergency originating restriction

CELF_CS_LINE_RESTRICT_DATA_ON	With originating restriction
CELF_CS_LINE_RESTRICT_DATA_OFF	Without originating restriction

0.1.38 Receive level change notification event structure

typedef struct{

int category; The value is VoiceNotify

int subtype; The value is VoiceNotify_RssiLevel

int info; Receive level

int subinfo; Line type

union {

...

} data ; // Unused

}CELF_CS_EVENT;

0.1.39 Receive level

CELF_CS_RSSI_LEVEL_0: Receive level 0

CELF_CS_RSSI_LEVEL_1: Receive level 1

CELF_CS_RSSI_LEVEL_2: Receive level 2

CELF_CS_RSSI_LEVEL_3: Receive level 3

0.1.40 line status struct

```
typedef struct  {
    unsigned char   LineStatus ;              Line status
    unsigned char   CoverageStatus ;          Area status information
    unsigned char   RRcmode ;                 RRC mode
    unsigned char   Network ;                 Network identification information
    unsigned char   unused[3] ;               unused
    unsigned char   ServiceStatus_AREA ;      Service status
    unsigned char   RestrictStatus ;          Restriction status
    unsigned char   NcRestriction ;           Normal originating restriction
    unsigned char   ServiceStatus_RES ;       Service status
    unsigned char   EcRestriction ;           Emergency originating restriction
} _CELF_CS_AREAREF_CHG_INF ;
```

0.1.41 Area status information

CELF_CS_LINE_CVR_STATUS_IN	IN
CELF_CS_LINE_CVR_STATUS_OUT	OUT

0.1.42 RRC mode

CELF_CS_LINE_RRC_MODE_IDLE	idle-mode
CELF_CS_LINE_RRC_MODE_UTRAN	utran-connected-mode

Network identification information

CELF_CS_LINE_NETWORK_HOME	home
CELF_CS_LINE_NETWORK_VISIT	visit
CELF_CS_LINE_NO_DATA	No data

0.1.43 Service status

CELF_CS_LINE_SRV_STATUS_CS	CS is in service.
CELF_CS_LINE_SRV_STATUS_PS	PS is in service.
CELF_CS_LINE_SRV_STATUS_CSPS	CS and PS are in service.
CELF_CS_LINE_NO_DATA	No data

CS is the circuit switched communication service, and

PS is the packet switched communication service.

0.1.44 Restriction status

CELF_CS_LINE_RESTRICT_ON	In traffic restriction
CELF_CS_LINE_RESTRICT_OFF	Out of traffic restriction

0.1.45 Supplementary service data structure

```
typedef struct  {
  unsigned char   Flg ;                              Identifying flag
 CELF_CS_NO_FLG : nothing
 CELF_CS_OPT_FLG : special number
 CELF_CS_USSD_FLG : USSD number
 char Title[CELF_SRVINFO_TITLE];         Supplementary service name  CELF_SRVINFO_TITLE=21
 char Send_no[CELF_SRVINFO_TITLE];   Dial data for accessing the service
                                                               CELF_SRVINFO_TITLE=40
} _CELF_CS_ADDSRV_DATA;
```

0.1.46 Response Message Data Structure

The supplementary response message information is the service name and Dial data, which is response message to send the network.

```
typedef struct {
    unsigned char    Title[ELIB_RESMSG_TITLE] ;          Service name
    unsigned char    Rcv_Msg[ELIB_RESMSG_DATA];      Dial data
} _CELF_CS_RESPONSE_MSG_DATA;
```

0.1.47 Date Format Structure

```
typedef struct {
unsigned char      Month
unsigned char      Day
unsigned char      Hour
unsigned char      Minute
} _CELF_MP_CS_DATE
```

1. Start Notification

1.1 Symbol: celf_mp_cs_notification_start

1.1.1 Syntax

```
celfMpStatus celf_mp_cs_notification_start (
        celfMpAppId      app_id,
        celfMpNotifySet  event_set,
        celfMpCallback   callback_func);
```

1.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsNotifySet

I/O: I

Description:

Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class **may** be registered to have a callback function called when the event occurs for the application identified by app_id. Classes of events are enabled by setting the corresponding bit in event_set:

The event classes are defined as follows:

CELF_MP_CS_CLASS_COM_STATUS:	Voice communication status notification
CELF_MP_CS_CLASS_TLK_TIME :	Call duration notification
CELF_MP_CS_CLASS_DISC_CAUSE:	Disconnection cause notification
CELF_MP_CS_CLASS_FW_RESULT :	Call forwarding result notification
CELF_MP_CS_CLASS_OFFHK_TO :	Off-hook originating timeout notification

A callback **may** be registered for all classes of events using special event class CELF_MP_CS_CLASS_ALL, however to reduce overhead it is recommended that only the needed event classes **should** be registered.

Name: callback_func

Type: celfMPCallback

I/O: I

Description:

The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

1.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_notification_start()` **shall** return one of the following values:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_EVENT_SET_ERR:	Notification event set is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

1.1.4 Include File

`/usr/include/celf/mp_cs.h`

1.1.5 Functional Description

This function is used to start notification callbacks for events related to circuit switched communication.

Events from a registered class **shall** cause the registered callback function to be called when the event occurs for the application identified by app_id. If a class of events does not have a registered callback function, no callback **shall** occur for those events.

Note: For further information about the event structure consult section 0.1 in this document.

2. Stop Notification

2.1 Symbol: celf_mp_cs_notification_stop

2.1.1 Syntax

```
celfMpStatus celf_mp_cs_notification_stop (
        celfMpAppId      app_id,
        celfMpNotifySet  event_set);
```

2.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsNotifySet

I/O: I

Description:

Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class **may** be registered to have a callback function called when the event occurs for the application identified by app_id. Classes of events are enabled by setting the corresponding bit in event_set:

The event classes are defined as follows:

CELF_MP_CS_CLASS_COM_STATUS:	Voice communication status notification
CELF_MP_CS_CLASS_TLK_TIME :	Call duration notification
CELF_MP_CS_CLASS_DISC_CAUSE:	Disconnection cause notification
CELF_MP_CS_CLASS_FW_RESULT :	Call forwarding result notification
CELF_MP_CS_CLASS_OFFHK_TO :	Off-hook originating timeout notification

A callback **may** be registered for all classes of events using special event class CELF_MP_CS_CLASS_ALL, however to reduce overhead it is recommended that only the needed event classes **should** be registered.

2.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_notification_stop()` **shall** return one of the following values:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_EVENT_SET_ERR:	Notification event set is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

2.1.4 Include File

`/usr/include/celf/mp_cs.h`

2.1.5 Functional Description

This function stops voice communication related event reporting.

For notification events, see "Start notification”.

Note: For further information about the event structure consult section 0.1 in this document.

3. Get Voice Communication Status

3.1 Symbol: celf_mp_cs_get_com_status

3.1.1 Syntax

celfMpStatus celf_mp_cs_get_com_status (

    celfMpAppId   app_id);

3.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

3.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_get_com_status()` **shall** return one of the values defined in section 0.1.

3.1.4 Include File

`/usr/include/celf/mp_cs.h`

3.1.5 Functional Description

This function gets the current voice communication status.

Without the monitoring the voice communication, it is possible to get the status of voice communication.

4. Get Connection Information to Other Party

4.1 Symbol: celf_mp_cs_get_con_info_ref

4.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_con_info_ref (
        celfMpAppId             app_id,
        celfMpCallNo            call_no,
        celfMPConnectInfo       connect_inf_p);
```

4.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: call_no

Type: celfMpCallNo

I/O: I

Description:

Call reference (0 to 255).

Name: connect_inf_p

Type: celfMPConnectInfo

I/O: O

Description:

Pointer to the connection destination information. See section 0.1 for details.

4.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_get_con_info_ref()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_CALL_NO_ERR:	Call number is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

4.1.4 Include File

`/usr/include/celf/mp_cs.h`

4.1.5 Functional Description

This function refers to the connection information to other party specified call reference

Without the monitoring the voice communication, it is possible to get the connection information

In the following cases, The result (STS) is set CELF_CS_ERR.

1. The call specified by call reference does not exist.
2. Others parameter Error.

5. Get Call Duration

5.1 Symbol: celf_mp_cs_get_call_duration

5.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_call_duration (
        celfMpAppId     app_id);
```

5.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

5.1.3 Return Value

Type: celfMpTime

I/O: O

Description:

`celf_mp_cs_get_call_duration()` **shall** return the current call duration in seconds.

5.1.4 Include File

`/usr/include/celf/mp_cs.h`

5.1.5 Functional Description

This function gets the call duration on the current call.

The call duration is counted by the voice communication service.

When no call exists, the function returns zero.

6. Off-Hook Notification

6.1 Symbol: celf_mp_cs_notification_off_hook

6.1.1 Syntax

```
celfMpStatus celf_mp_cs_notification_off_hook (
        celfMpAppId      app_id,
        celfMpCsBtype    com_type,
        celfMpCsOffHk    option);
```

6.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name:	com_type
Type:	celfCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

Name:	option
Type:	celfCsOffHk
I/O:	I

Description:

One the following options **shall** be set:

CELF_CS_OFFHK_AUTO Automatic transmission

CELF_CS_OFFHK_MANUAL Manual transmission

6.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_notification_off_hook()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.

CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

6.1.4 Include File

`/usr/include/celf/mp_cs.h`

6.1.5 Functional Description

This function receives the request of off-hook.

By this function,

(1) When the mobile phone is in the wait (standby) status, the dial tone (DT) sounds and it is possible to input dial number, or

(2) When the input of dial number is completed, the mobile phone starts the originating.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The process at timer timeout (five seconds) varies depending on the specification of “option”.

This timer count starts at the last dial inputting.

(1) When the "option" is CELF_CS_OFFHK_AUTO (automatic originating)

Automatic originating operation is immediately performed by the dials, which were already input in "Dial ".

(2) When the "option" is CELF_CS_OFFHK_MANUAL (manual originating)

It is notified timeout to an application, and waits for the notification of originating from

the application. ("Complete dial" or "On-hook originating")

Timeout is notified by monitoring "Off-hook originating timeout notification" in "Start

voice communication status monitoring”.

When a mobile phone is moved to low voltage mode, a low voltage notification is sent.

During low voltage, when the communication status is other than the under standby, this Off-hook is disabled.

If an incoming call arrives during off-hook, this Off-hook is cancelled.

In case of using the subaddress, it should be use the function "On-hook originating".

7. Disconnect

7.1 Symbol: celf_mp_cs_disconnect

7.1.1 Syntax

```
celfMpStatus celf_mp_cs_disconnect (
        celfMpAppId     app_id
        celfMpCsBtype   com_type);
```

7.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: com_type

Type: celfMpCsBtype

I/O: I

Description:

Communication type as defined in section 0.1.

7.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_disconnect()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

7.1.4 Include File

`/usr/include/celf/mp_cs.h`

7.1.5 Functional Description

This function receives the request to disconnect the call.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

An incoming call cannot be disconnected by this function. (Use "Reject incoming call")

If multiple calls exist, all calls are disconnected.

8. Dial

8.1 Symbol: celf_mp_cs_dial

8.1.1 Syntax

```
celfMpStatus celf_mp_cs_dial (
        celfMpAppId         app_id
        celfMpCsBtype       com_type,
        celfMpCsDialBuffer          dial_buf,
        celfMpCsDialLen             dial_len);
```

8.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name:	com_type
Type:	celfMpCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

Name:	dial_buf
Type:	celfMpCsDialBuffer
I/O:	I

Description:

Dial data buffer address

Name:	dial_len
Type:	celfMpCsDialLen
I/O:	I

Description:

Dial data length

8.1.3 Return Value

Type:	celfMpStatus

I/O: O

Description:

`celf_mp_cs_dial()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

8.1.4 Include File

`/usr/include/celf/mp_cs.h`

8.1.5 Functional Description

This function receives the sequence of dial number.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The dial data stores the following ASCII codes.

1 : 0 x 31	2 : 0 x 32	3 : 0 x 33
4 : 0 x 34	5 : 0 x 35	6 : 0 x 36
7 : 0 x 37	8 : 0 x 38	9 : 0 x 39
* : 0 x 2a	0 : 0 x 30	# : 0 x 23

When "Off-hook" is called, the mobile phone is in off-hook status.

Under this off-hook status, the mobile phone starts outgoing with "Dial" and "Complete dial".

Five seconds later from the last dialling, the outgoing process starts automatically, when automatic transmission is specified in "Off-hook".

When "Off-hook" is called, the mobile phone is in off-hook status.

Under this on-hook status, DTMF is sent, if the status is (a) the conversation or (b) the conversation and hold.

9. Dial Complete

9.1 Symbol: celf_mp_cs_dial_end

9.1.1 Syntax

```
celfMpStatus celf_mp_cs_dial_end (
        celfMpAppId     app_id
        celfMpCsBtype   com_type);
```

9.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: com_type

Type: celfMpCsBtype

I/O: I

Description:

Communication type as defined in section 0.1.

9.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_dial_end()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

9.1.4 Include File

`/usr/include/celf/mp_cs.h`

9.1.5 Functional Description

This function receives the request to end of inputting dials.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

Under off-hook status, the mobile phone starts outgoing operation by calling this function with dial number, which was input by preceding function calls "Dial".

Under on-hook status, the calling this function is disabled.

10.Response to Incoming Call

10.1 Symbol: celf_mp_cs_call_rcv

10.1.1 Syntax

```
celfMpStatus celf_mp_cs_call_rcv (
        celfMpAppId     app_id
        celfMpCsBtype   com_type);
```

10.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: com_type

Type: celfMpCsBtype

I/O: I

Description:

Communication type as defined in section 0.1.

10.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_call_rcv()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

10.1.4 Include File

`/usr/include/celf/mp_cs.h`

10.1.5 Functional Description

This function receives the request to process an incoming call.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

One of the following operations is performed depending on the mobile phone status.

Under incoming	:	Responds to the incoming call.
Under response hold	:	Responds to the response hold call
Others	:	Disabled

The mobile phone is in low voltage mode, this function is disabled.

To respond to the incoming call in the status, “under conversation and incomings”, use "Reject incoming call".

11.Forward Incoming Call

11.1 Symbol: celf_mp_cs_call_forward

11.1.1 Syntax

```
celfMpStatus celf_mp_cs_call_forward (
        celfMpCsBtype   com_type);
```

11.1.2 Argument

Name:	com_type
Type:	celfMpCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

11.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_call_forward()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

11.1.4 Include File

`/usr/include/celf/mp_cs.h`

11.1.5 Functional Description

This function receives the request to forwarding an incoming call.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The incoming call is forwarded when the communication status is (a)under the incoming, (b)under conversation and incoming, or (c)under hold and incoming.

If the forwarding fails, incoming call is continued between other party and this phone.

12.Forward to Phone Answering Message

12.1 Symbol: celf_mp_cs_call_forward_voice_msg

12.1.1 Syntax

```
celfMpStatus celf_mp_cs_call_forward_voice_msg (
        celfMpCsBtype   com_type);
```

12.1.2 Argument

Name:	com_type
Type:	celfMpCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

12.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_call_forward_voice_msg()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

12.1.4 Include File

`/usr/include/celf/mp_cs.h`

12.1.5 Functional Description

This function receives the request to forward a call to a phone-answering message.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The incoming call is forwarded to phone-answering message when the communication status is (a) under the incoming, (b) under conversation and incoming, or (c) under hold and incoming.

If the forwarding fails, incoming call is continued between other party and this phone.

13.Call Hold

13.1 Symbol: celf_mp_cs_call_hold

13.1.1 Syntax

celfMpStatus celf_mp_cs_call_hold (

celfMpCsBtype com_type);

13.1.2 Argument

Name:	com_type
Type:	celfMpCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

13.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_call_hold()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

13.1.4 Include File

`/usr/include/celf/mp_cs.h`

13.1.5 Functional Description

This function receives the requests response hold.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

This response hold is performed for an incoming call, only when the communication status is under incoming.

To release response hold (move to the under conversation status) call "Response to an incoming call".

14.Call Reject

14.1 Symbol: celf_mp_cs_call_reject

14.1.1 Syntax

```
celfMpStatus celf_mp_cs_call_reject (
        celfMpCsBtype   com_type);
```

14.1.2 Argument

Name:	com_type
Type:	celfMpCsBtype
I/O:	I

Description:

Communication type as defined in section 0.1.

14.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_call_reject()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

14.1.4 Include File

`/usr/include/celf/mp_cs.h`

14.1.5 Functional Description

This function receives the request to reject an incoming call.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The operation for each communication status is as follows:

Under incoming:	Rejects an incoming call
Under conversation and incoming:	Rejects an incoming call

Under hold and incoming:	Rejects an incoming call
Under conversation, hold, and incoming:	Rejects an incoming call

15.Multi Party Call

15.1 Symbol: celf_mp_cs_mp_call

15.1.1 Syntax

```
celfMpStatus celf_mp_cs_mp_call (
        celfMpCsBtype   com_type,
        celfMpCsMop     kind,
        celfMpCsCallRef cr);
```

15.1.2 Argument

Name: com_type

Type: celfMpCsBtype

I/O: I

Description:

Communication type as defined in section 0.1.

Name: kind

Type: celfMpCsMop

I/O: I

Description:

Operation type

CELF_CS_MOP_RSV_DISC:	Disconnect the hold call
CELF_CS_MOP_DISC_AND_RSP:	Response after disconnection
CELF_CS_MOP_RSV_AND_RSP:	Response after hold (including operation for switching a call)
CELF_CS_MOP_CR_DISC:	Disconnect call with specific call reference

Name: cr

Type: celfMpCsCallRef

I/O: I

Description:

Call reference of the call to be disconnected

Valid only if CELF_CS_MOP_CR_DISC is specified for the second argument.

15.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_mp_call()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

15.1.4 Include File

`/usr/include/celf/mp_cs.h`

15.1.5 Functional Description

This function receives the request to operate for each call, when communication is made with multiple calls.

The operation is as follows depending on ìTypeî:

- CELF_CS_MOP_RSV_DISC

If a hold call exists, this hold call is disconnected.

- CELF_CS_MOP_DISC_AND_RSP

If a conversation call exists and if another call status is incoming or hold,

the conversation call transits to disconnect status and another call transits to conversation status.

See detail below.

(1) Under conversation and incoming

This status is that 1st call is conversation, and 2nd call is incoming.

The result is that 1st call is release, and 2nd call is conversation.

(2) Under conversation and hold

This status is that 1st call is conversation, and 2nd call is hold.

The result is that 1st call is release, and 2nd call is conversation.

(3) Under conversation, hold, and incoming

This status is that 1st call is conversation, that 2nd call is hold, and that 3rd call is incoming.

The result is that 1st call is release, that 2nd call maintains hold, and that 3rd call is conversation.

(4) Under response hold

This status is not changed.

- CELF_CS_MOP_RSV_AND_RSP

If a conversation call exists and if another call status is incoming or hold, the conversation call transits to hold status and another call transits to conversation status.

See detail below.

(1) Under conversation and incoming

This status is that 1st call is conversation, and 2nd call is incoming.

The result is that 1st call is hold, and 2nd call is conversation.

(2) Under conversation and hold

This status is that 1st call is conversation, and 2nd call is hold.

The result is that 1st call is hold, and 2nd call is conversation.

(3) Under conversation, hold, and incoming

This status is that 1st call is conversation, that 2nd call is hold, and that 3rd call is incoming.

The result is that 1st call is hold, that 2nd call is conversation, that 3rd call maintains incoming.

(4) Under response hold

This status is that 1st call is hold.

The result is that 1st call is conversation.

- CELF_CS_MOP_CR_DISC It is disconnect the call specified the call reference.

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

16.On-Hook Originating

16.1 Symbol: celf_mp_cs_originating_on_hook

16.1.1 Syntax

celfMpStatus celf_mp_cs_originating_on_hook (

celfMpAppId	app_id,
celfMpCsConReq	con_req);

16.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: con_req

Type: celfMpCsConReq

I/O: I

Description:

Communication request type as defined in section 0.1.

16.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_call_reject()` **shall** return one of the values defined:

CELF_CS_OK:	Normal
CELF_CS_ONHOOK_DENY:	On-hook originating is impossible.
CELF_CS_ONHOOK_STATUS_ERR:	Error due to communication conflict
CELF_CS_ONHOOK_OB_CR:	Excess of the maximum number of calls
CELF_CS_ERR:	Abnormal

16.1.4 Include File

`/usr/include/celf/mp_cs.h`

16.1.5 Functional Description

This function receives the request to originate to the specified dial number.

The communication status should be Standby.

The dial number is specified by "dial_buf" and "subaddr_buf" in the “con_req” structure.

If the character string, "184" or "186", is placed at the head of dial data, this character string is deleted.

Whether the originating dial number is notified or not, it is identified by "notice".

The dial data and subaddress stores the following ASCII codes.

1 : 0 x 31	2 : 0 x 32	3 : 0 x 33
4 : 0 x 34	5 : 0 x 35	6 : 0 x 36
7 : 0 x 37	8 : 0 x 38	9 : 0 x 39
* : 0 x 2a	0 : 0 x 30	# : 0 x 23

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "Start voice communication status monitoring" to obtain the communication status.

The originating request during low voltage is disabled.

17.Get Call Reference

17.1 Symbol: celf_mp_cs_get_call_reference

17.1.1 Syntax

celfMpStatus celf_mp_cs_get_call_reference (

celfMpAppId app_id

celfMpCsChanNum channel_num);

17.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name:	channel_num
Type:	celfMpCsChanNum
I/O:	O

Description:

Channel number information as defined in section 0.1.

17.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_get_call_reference()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

17.1.4 Include File

`/usr/include/celf/mp_cs.h`

17.1.5 Functional Description

This function gets the call reference in use.

A value within 0 to 255 is set to "ChanNum_00", "ChanNum_01" and "ChanNum_02". If channel is not used, CELF_CS_CHAN_NOUSE is set as the call reference.

Three channel corresponds to three call in multiple call.

18.Start DCF message monitoring

18.1 Symbol: celf_mp_cs_DCF_monitoring_start

18.1.1 Syntax

```
celfMpStatus celf_mp_cs_DCF_monitoring_start (
        celfMpAppId     app_id
        celfMpDCFSet    event_set);
```

18.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsDCFSet

I/O: I

Description:

Notification event set. Events that are classified as belonging to one of the `CelfMpCsDCFSet` class **may** be registered to have a callback function called when the event occurs for the application identified by app_id. Classes of events are enabled by setting the corresponding bit in event_set:

The event classes are defined as follows:

CELF_MP_CS_DCF_DISP	Display-related message
CELF_MP_CS_DCF_HISTORY	History-related message
CELF_MP_CS_DCF_TONE1	Tone 1-related message
CELF_MP_CS_DCF_TONE2	Tone 2-related message
CELF_MP_CS_DCF_ETC	Other messages
CELF_MP_CS_CLASS_ALL	All notified

A callback **may** be registered for all classes of events using special event class CELF_MP_CS_CLASS_ALL, however to reduce overhead it is recommended that only the needed event classes **should** be registered.

Name: callback_func

Type: celfMPCallback

I/O: I

Description:

The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

18.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_DCF_monitoring_start ()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

18.1.4 Include File

`/usr/include/celf/mp_cs.h`

18.1.5 Functional Description

This function starts the monitoring the DCF message on the voice communication or AV communication.

The occurrence of the event is notified to the application, specified by app_id.

The messages to be notified are described below.

Display-related message:

-Notification of starting display during CCP outgoing

-Notification of starting display during CCP incoming

-Notification of starting display during CCP calling

-Notification of starting display during CCP connecting

-Notification of starting display during CCP communication

-Notification of ending CCP That is to notifies of release of a CCP call.

-Notification of starting CCP disconnection (on the mobilephone) display

-Notification of starting display of CCP disconnection (on the network) display

-Notification of rejecting CCP outgoing

-Notification of CCP hold

-Notification of releasing CCP hold

History-related message:

-Notification of registering CCP outgoing call history

-Notification of registering CCP absence incoming call history

-Notification of registering CCP incoming call history

Tone 1-related message:(Tone sounding on the AP layer)

-Notification of CCP RGT start

-Notification of CCP RGT stop

-Start report of incoming of a CCP hold call

-Stop report of incoming of a CCP hold call

Tone 2-related message:(Tone sounding by the voice communication service)

-Notification of CCP DST start

-Notification of CCP DST stop

-Notification of CCP RBT start

-Notification of CCP RBT stop

-Notification of CCP BT start

-Notification of CCP CWT start

-Notification of CCP CWT stop

Other messages:

-Inquiry report of rejecting a CCP CS incoming call

19.Stop DCF message monitoring

19.1 Symbol: celf_mp_cs_DCF_monitoring_stop

19.1.1 Syntax

```
celfMpStatus celf_mp_cs_DCF_monitoring_stop (
        celfMpAppId     app_id
        celfMpDCFSet    event_set);
```

19.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsDCFSet

I/O: I

Description:

Notification event set. Events that are classified as belonging to one of the `CelfMpCsDCFSet` class. Classes of events are enabled by setting the corresponding bit in event_set:

The event classes are defined as follows:

CELF_MP_CS_DCF_DISP	Display-related message
CELF_MP_CS_DCF_HISTORY	History-related message
CELF_MP_CS_DCF_TONE1	Tone 1-related message
CELF_MP_CS_DCF_TONE2	Tone 2-related message
CELF_MP_CS_DCF_ETC	Other messages
CELF_MP_CS_CLASS_ALL	All notified

19.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_DCF_monitoring_stop()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion

CELF_MP_STATUS_ERR:	Other unsuccessful completion.

19.1.4 Include File

`/usr/include/celf/mp_cs.h`

19.1.5 Functional Description

This function stops notifying of the DCF message on voice communication or AV communication.

20.Voice Message Notification

20.1 Symbol: celf_mp_cs_voice_msg_notify

20.1.1 Syntax

celfMpStatus celf_mp_cs_voice_msg_notify (

celfMpCsRecMsg rec_status);

20.1.2 Argument

Name:	rec_status
Type:	celfMpCsRecMsg
I/O:	I

Description:

CELF_CS_REC_MESSAGE_START:Start of a voice message

CELF_CS_REC_MESSAGE_STOP:Stop of a voice message

20.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_call_voice_msg_notify()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

20.1.4 Include File

`/usr/include/celf/mp_cs.h`

20.1.5 Functional Description

Functional description

This function must be called before the communication state is changed to "under conversation."

After the start notification, the APL must issue the stop notification when the voice message is stopped.

21.Hold Tone Start

21.1 Symbol: celf_mp_cs_hold_tone_start

21.1.1 Syntax

```
celfMpStatus celf_mp_cs_hold_tone_start (
        celfMpAppId     app_id,);
```

21.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

21.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_hold_tone_start()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

21.1.4 Include File

`/usr/include/celf/mp_cs.h`

21.1.5 Functional Description

This function starts to sound a hold tone during a call.

22.Hold Tone Stop

22.1 Symbol: celf_mp_cs_hold_tone_stop

22.1.1 Syntax

```
celfMpStatus celf_mp_cs_hold_tone_stop (
        celfMpAppId     app_id);
```

22.1.2 Argument

Name:	app_id
Type:	celfMpAppId
I/O:	I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

22.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_hold_tone_stop()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_COM_TYPE_ERR:	Communication type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

22.1.4 Include File

`/usr/include/celf/mp_cs.h`

22.1.5 Functional Description

This function stops to sound a hold tone during a call.

23.Get 64K / AV Communication Status

23.1 Symbol: celf_mp_cs_get_UD_com_stat

23.1.1 Syntax

celfMpStatus celf_mp_cs_get_UD_com_stat (

void);

23.1.2 Argument

None.

23.1.3 Return Value

Type:	celfMpUDComStatus
I/O:	O

Description:

`celf_mp_cs_get_UD_com_stat()` **shall** return one of the values defined:

CELF_CS_UD_STOP: Under stop

CELF_CS_UD_RUN: Under communication

CELF_CS_UD_CALLED: Under incoming

CELF_CS_UD_CALLING: Under outgoing

CELF_CS_UD_DISCONNECT: Under disconnection

CELF_CS_UD_CALLING_ALERT: Under calling

CELF_CS_UD_HOLD: Under hold

23.1.4 Include File

`/usr/include/celf/mp_cs.h`

23.1.5 Functional Description

This function refers to the communication status of 64K communication or AV communication.

24.Get internal/external AV Communication Status

24.1 Symbol: celf_mp_cs_get_AV_com_stat

24.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_AV_com_stat (
        void);
```

24.1.2 Argument

None.

24.1.3 Return Value

Type:	celfMpAVComStatus
I/O:	O

Description:

`celf_mp_cs_get_AV_com_stat()` **shall** return one of the values defined:

CELF_CS_AV_IN_STOP: Under stop

CELF_CS_AV_IN_RUN: Under communication

CELF_CS_AV_IN_CALLED: Under incoming

CELF_CS_AV_IN_CALLING: Under outgoing

CELF_CS_AV_IN_DISCONNECT: Under disconnection

CELF_CS_AV_IN_CALLING_ALERT: Under calling

CELF_CS_UD_IN_HOLD: Under hold

CELF_CS_AV_OUT_STOP: Under stop

CELF_CS_AV_OUT_RUN: Under communication

CELF_CS_AV_OUT_CALLED: Under incoming

CELF_CS_AV_OUT_CALLING: Under outgoing

CELF_CS_AV_OUT_DISCONNECT: Under disconnection

CELF_CS_AV_OUT_CALLING_ALERT: Under calling

CELF_CS_UD_OUT_HOLD: Under hold

24.1.4 Include File

`/usr/include/celf/mp_cs.h`

24.1.5 Functional Description

This function refers to the communication status of internal or external AV communication.

25.Get Communication Status

25.1 Symbol: celf_mp_cs_get_com_stat

25.1.1 Syntax

celfMpStatus celf_mp_cs_get_com_stat (

celfMpAppId app_id,

celfMpCsRcvScene rcv_scene_p);

25.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: rcv_scene_p

Type: celfMpCsRcvScene

I/O: O

Description:

Incoming call type:

CELF_CS_RCV_SCENE_COMPETE_TRN : Outgoing conflict

CELF_CS_RCV_SCENE_RSV_RETURN : Incoming hold call

CELF_CS_RCV_SCENE_CALL_BACK : Re-incoming

CELF_CS_RCV_SCENE_NORMAL: Normal

CELF_CS_RCV_SCENE_NON: Unset

25.1.3 Return Value

Type: celfMpComStatus

I/O: O

Description:

`celf_mp_cs_get_com_stat()` **shall** return one of the values defined:

Current communication status

CELF_CS_COM_STATUS_WAIT:	Standby
CELF_CS_COM_STATUS_RCV:	Under incoming
CELF_CS_COM_STATUS_TRN:	Under outgoing
CELF_CS_COM_STATUS_DLV:	Under calling
CELF_CS_COM_STATUS_TLK:	Under conversation
CELF_CS_COM_STATUS_HLD:	Under response hold

CELF_CS_COM_STATUS_DUMMY1:	Under off-hook
CELF_CS_COM_STATUS_RLS:	Under release
CELF_CS_COM_STATUS_TLK_RCV:	Under conversation and incoming
CELF_CS_COM_STATUS_TLK_TRN:	Under conversation and outgoing
CELF_CS_COM_STATUS_TLK_DLV:	Under conversation and calling
CELF_CS_COM_STATUS_TLK_RSV:	Under conversation and hold
CELF_CS_COM_STATUS_TLK_RLS:	Under conversation and release
CELF_CS_COM_STATUS_TLK_RSV_RCV:	Under conversation, hold, and incoming
CELF_CS_COM_STATUS_RCV_AV:	Under incoming of an AV call
CELF_CS_COM_STATUS_TRN_AV:	Under outgoing of an AV call
CELF_CS_COM_STATUS_DLV_AV:	Under calling of an AV call
CELF_CS_COM_STATUS_TLK_AV:	Under conversation of an AV call
CELF_CS_COM_STATUS_HLD_AV:	Under response hold of an AV call
CELF_CS_COM_STATUS_RLS_AV:	Under release of an AV call
CELF_CS_COM_STATUS_DUMMY2 :	Under AV off-hook
CELF_CS_ERR :	Abnormal end

25.1.4 Include File

`/usr/include/celf/mp_cs.h`

25.1.5 Functional Description

This function returns the incoming call status, when the current call is (a) under incoming status or (b) under conversation and incoming status.

26.Start Line Status Monitoring

26.1 Symbol: celf_mp_cs_monitor_start

26.1.1 Syntax

celfMpStatus celf_mp_cs_monitor_start (

celfMpAppId	app_id
celfMpCsMtype	event_set,
celfMpCallback	callback_func);

26.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsMtype

I/O: I

Description:

Events with bits on are notified.

CELF_CS_MONITOR_LINE_STATUS:	Line status change notification
CELF_CS_MONITOR_RESTRICT:	Restriction status change notification
CELF_CS_MONITOR_RSSI:	Receive level change notification
CELF_CS_MONITOR_ALL:	All notified

Name: callback_func

Type: celfMPCallback

I/O: I

Description:

The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

26.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_monitor_start()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_MON_TYPE_ERR:	Monitor type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

26.1.4 Include File

`/usr/include/celf/mp_cs.h`

26.1.5 Functional Description

This function starts the monitoring the line status.

The occurrence of the event is notified to the application, specified by app_id.

The events to be notified are described below.

1. Line status change notification:

This event notifies that the line status is changed.

The line status is the out-of-communication area status and the within-communication area.

2. Restriction status change notification:

This event notifies that a restriction status is changed.

The restriction means that the incoming call or the outgoing call is restricted by the network in case of traffic congestion.

3. Receive level change notification:

This event notifies that the receive level is changed.

The receive revel is the intensity of electromagnetic wave. The intensity is four revel, high, mid, low and zero (out of area).

See section 0.1 for structure definitions and values.

27.Stop Line Status Monitoring

27.1 Symbol: celf_mp_cs_monitor_stop

27.1.1 Syntax

```
celfMpStatus celf_mp_cs_monitor_stop (
        celfMpAppId     app_id
        celfMpCsMtype   event_set);
```

27.1.2 Argument

Name: app_id

Type: celfMpAppId

I/O: I

Description:

Application ID returned from `celf_mp_af_get_app_id()` call.

Name: event_set

Type: celfMpCsMtype

I/O: I

Description:

Mask of the events for which reporting is to be stopped.

Formed by the bitwise OR of one or more of the following line status related events:

CELF_CS_MONITOR_LINE_STATUS:	Line status change notification
CELF_CS_MONITOR_RESTRICT:	Restriction status change notification
CELF_CS_MONITOR_RSSI:	Receive level change notification
CELF_CS_MONITOR_ALL:	All notified

27.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_monitor_stop()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_APP_ID_ERR:	Application ID is not valid.
CELF_MP_STATUS_MON_TYPE_ERR:	Monitor type is not valid
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

27.1.4 Include File

```
/usr/include/celf/mp_cs.h
```

27.1.5 Functional Description

This function ends notifying on the event of the line status.

28.Get Receive Level

28.1 Symbol: celf_mp_cs_get_rcv_level

28.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_rcv_level (
        void);
```

28.1.2 Argument

None.

28.1.3 Return Value

Type:	celfMpRcvLevel
I/O:	O

Description:

`celf_mp_cs_get_rcv_level()` **shall** return one of the values defined:

CELF_CS_RSSI_LEVEL_0: Receive level 0

CELF_CS_RSSI_LEVEL_1: Receive level 1

CELF_CS_RSSI_LEVEL_2: Receive level 2

CELF_CS_RSSI_LEVEL_3: Receive level 3

28.1.4 Include File

`/usr/include/celf/mp_cs.h`

28.1.5 Functional Description

This function obtains the current receive level.

Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the status of receive level.

29.Get Line Status

29.1 Symbol: celf_mp_cs_get_line_status

29.1.1 Syntax

celfMpStatus celf_mp_cs_get_line_status (

CELF_CS_AREAREF_CHG_INF * net);

29.1.2 Argument

Name:	net
Type:	CELF_CS_AREAREF_CHG_INF
I/O:	I

Description:

Pointer to the struct used to hold line status information

29.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_get_line_status()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

29.1.4 Include File

`/usr/include/celf/mp_cs.h`

29.1.5 Functional Description

This function obtains the current line status.

Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the status of line status.

See section 0.1 for further information.

30.Get Coverage Status

30.1 Symbol: celf_mp_cs_get_coverage_status

30.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_line_status (
        _CELF_CS_LINE_STATUS_EX *        net,
        celfMpCsCoverage        cover);
```

30.1.2 Argument

Name: net

Type: _CELF_CS_LINE_STATUS_EX

I/O: I

Description:

Pointer to the struct used to hold line status information

Name: cover

Type: celfMpCsCoverage

I/O: I

Description:

Within- or out-of communication area status

CELF_CS_LINE_STATUS_IN: Within-communication area

CELF_CS_LINE_STATUS_OUT: Out-of-communication area

30.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_get_line_status()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

30.1.4 Include File

`/usr/include/celf/mp_cs.h`

30.1.5 Functional Description

This function obtains the information on the current status of the within- and out-of-communication areas for current line.

(This function gets only information of inside or outside coverage area status.)

31.Get Voice Mail Information

31.1 Symbol: celf_mp_cs_get_vm_info

31.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_vm_info (
        celfMpCsVMNum *     vm_num_p);
```

31.1.2 Argument

Name:	vm_num_p
Type:	celfMpCsVMNum
I/O:	I

Description:

Address of the storage area of the number of stored phone-answering messages

31.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_get_vm_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

31.1.4 Include File

`/usr/include/celf/mp_cs.h`

31.1.5 Functional Description

This function obtains the storage status of phone-answering messages from nonvolatile memory.

The storage status is the number of message of phone-answering.

32.Set Voice Mail Information

32.1 Symbol: celf_mp_cs_set_vm_info

32.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_vm_info (
        celfMpCsVMNum        vm_num);
```

32.1.2 Argument

Name: vm_num

Type: celfMpCsVMNum

I/O: I

Description:

The number of stored phone-answering messages

32.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_vm_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

32.1.4 Include File

`/usr/include/celf/mp_cs.h`

32.1.5 Functional Description

This function sets the storage status of phone-answering message to non-volatile memory.

33.Get Call Selection

33.1 Symbol: celf_mp_cs_get_call_select

33.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_call_select (
        void);
```

33.1.2 Argument

None.

33.1.3 Return Value

Type: celfMpCallSelect

I/O: O

Description:

CELF_CS_INCOMING_VOICE_ANSWERING:	Forward to the phone-answering message
CELF_CS_INCOMING_FORWARD:	Forward
CELF_CS_INCOMING_REJECT:	Reject (disconnect)
CELF_CS_INCOMING_NORMAL:	Receipt of an incoming call (normal incoming)

33.1.4 Include File

`/usr/include/celf/mp_cs.h`

33.1.5 Functional Description

This function obtains the incoming call information from non-volatile memory.

Refer "Set incoming function selection"

34.Set Call Selection

34.1 Symbol: celf_mp_cs_set_call_select

34.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_call_select (
        celfMpCallSelect select);
```

34.1.2 Argument

Name:	select
Type:	celfMpCallSelect
I/O:	I

Description:

CELF_CS_INCOMING_VOICE_ANSWERING:	Forward to the phone-answering message
CELF_CS_INCOMING_FORWARD:	Forward
CELF_CS_INCOMING_REJECT:	Reject (disconnect)
CELF_CS_INCOMING_NORMAL:	Receipt of an incoming call (normal incoming)

34.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_call_select()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

34.1.4 Include File

`/usr/include/celf/mp_cs.h`

34.1.5 Functional Description

This function sets the incoming call information to nonvolatile memory.

When an incoming call arrives during conversation mode, it is possible to save this incoming call information.

35.Set Service Information

35.1 Symbol: celf_mp_cs_set_service_info

35.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_service_info (
        celfMpRegNum   reg_no,
        celfMpCsSrvData *         data_p);
```

35.1.2 Argument

Name:	reg_no
Type:	celfMpRegNum
I/O:	I

Description:

Registration number: 1 to 10

Name:	data_p
Type:	celfMpCsSrvData
I/O:	I

Description:

Pointer to supplementary service data

35.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_service_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

35.1.4 Include File

`/usr/include/celf/mp_cs.h`

35.1.5 Functional Description

This function registers the supplementary service information to the nonvolatile memory,

The supplementary service information is the service name and Dial data for accessing the service. The ìdial_noî is used as the key for accessing this supplementary service.

The value range is from 0 to 10.

See section 0.1 for additional information.

36.Get Service Information

36.1 Symbol: celf_mp_cs_get_service_info

36.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_service_info (
            celfMpRegNum   reg_no,
            celfMpCsSrvData *          data_p);
```

36.1.2 Argument

Name: reg_no

Type: celfMpRegNum

I/O: I

Description:

Registration number: 1 to 10

Name: data_p

Type: celfMpCsSrvData

I/O: I

Description:

Pointer to supplementary service data

36.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_get_service_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

36.1.4 Include File

`/usr/include/celf/mp_cs.h`

36.1.5 Functional Description

This function obtains supplementary service information, specified by “reg_no”, from non-volatile memory.

See “Register supplementary service settings”.

37.Delete Service Information

37.1 Symbol: celf_mp_cs_del_service_info

37.1.1 Syntax

```
celfMpStatus celf_mp_cs_del_service_info (
        celfMpRegNum   reg_no);
```

37.1.2 Argument

Name:	reg_no
Type:	celfMpRegNum
I/O:	I

Description:

Registration number: 1 to 10

37.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_del_service_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

37.1.4 Include File

`/usr/include/celf/mp_cs.h`

37.1.5 Functional Description

This function deletes the supplementary service information specified by “reg_no” from non-volatile memory.

38.Remove Service Information

38.1 Symbol: celf_mp_cs_remove_all_service_info

38.1.1 Syntax

```
celfMpStatus celf_mp_cs_remove_all_service_info (
        void);
```

38.1.2 Argument

None.

38.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_remove_all_service_info()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

38.1.4 Include File

`/usr/include/celf/mp_cs.h`

38.1.5 Functional Description

This function deletes all the supplementary service information from non-volatile memory.

39.Set Response Message Settings

39.1 Symbol: celf_mp_cs_set_resp_msg

39.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_resp_msg (
        celfMpRegNum   reg_no,
        celfMpCsSrvData *       data_p);
```

39.1.2 Argument

Name:	reg_no
Type:	celfMpRegNum
I/O:	I

Description:

Registration number: 1 to 10

Name:	data_p
Type:	celfMpCsMsgData
I/O:	I

Description:

Pointer to the additional response message setting data area

39.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_resp_msg()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

39.1.4 Include File

`/usr/include/celf/mp_cs.h`

39.1.5 Functional Description

This function registers the supplementary response message information for the supplementary service to the non-volatile memory,

When a supplementary service is activated, and corresponding message from the network is received, this supplementary response message is sent to the network.

The supplementary response message information is the service name and Dial data, which is response message to send the network.

The dial data should be USSD.

The “reg_no” is used as the key for accessing this supplementary response message .

The value range is from 0 to 10.

For information about the structures, see section 0.1.

40.Get Response Message Settings

40.1 Symbol: celf_mp_cs_get_resp_msg

40.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_resp_msg (
        celfMpRegNum   reg_no,
        celfMpCsSrvData *        data_p);
```

40.1.2 Argument

Name:	reg_no
Type:	celfMpRegNum
I/O:	I

Description:

Registration number: 1 to 10

Name:	data_p
Type:	celfMpCsMsgData
I/O:	I

Description:

Pointer to the additional response message setting data area

40.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_get_resp_msg()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

40.1.4 Include File

`/usr/include/celf/mp_cs.h`

40.1.5 Functional Description

This function obtains the supplementary response message information, specified by “reg_no”, from non-volatile memory.

See “Register response message settings”.

41.Delete Response Message Settings

41.1 Symbol: celf_mp_cs_del_resp_msg

41.1.1 Syntax

```
celfMpStatus celf_mp_cs_del_resp_msg (
        celfMpRegNum   reg_no);
```

41.1.2 Argument

Name:	reg_no
Type:	celfMpRegNum
I/O:	I

Description:

Registration number: 1 to 10

41.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_del_resp_msg()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

41.1.4 Include File

`/usr/include/celf/mp_cs.h`

41.1.5 Functional Description

This function deletes the supplementary response message information, specified by “reg_no”, from non-volatile memory.

42.Remove All Response Message Settings

42.1 Symbol: celf_mp_cs_remove_all_resp_msg

42.1.1 Syntax

celfMpStatus celf_mp_cs_remove all_resp_msg (

void);

42.1.2 Argument

None.

42.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_remove_all_resp_msg()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

42.1.4 Include File

`/usr/include/celf/mp_cs.h`

42.1.5 Functional Description

This function removes all the supplementary response message information, specified by “reg_no”, from non-volatile memory.

43.Set Reconnection Tone

43.1 Symbol: celf_mp_cs_set_reconnection_tone

43.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_reconnection_tone (
        celfMpCsReconnectionTone        reconn);
```

43.1.2 Argument

Name:	reconn
Type:	celfMpCsReconnectionTone
I/O:	I

Description:

Reconnection tone to be set

CELF_CS_RECONN_ON_T_OFF:	Tone OFF
CELF_CS_RECONN_ON_T_LOW:	Tone ON low tone
CELF_CS_RECONN_ON_T_HI:	Tone ON high tone

43.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_reconnection_tone()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

43.1.4 Include File

`/usr/include/celf/mp_cs.h`

43.1.5 Functional Description

This function sets the reconnection tone information to the non-volatile memory.

The type of reconnection tone is specified by “reconn”

44.Get Reconnection Tone

44.1 Symbol: celf_mp_cs_get_reconnection_tone

44.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_reconnection_tone (
        void);
```

44.1.2 Argument

None.

44.1.3 Return Value

Type:	celfMpCsReconnectionTone
I/O:	O

Description:

`celf_mp_cs_get_reconnection_tone()` **shall** return one of the values defined:

CELF_CS_RECONN_ON_T_OFF: Tone OFF

CELF_CS_RECONN_ON_T_LOW: Tone ON low tone

CELF_CS_RECONN_ON_T_HI: Tone ON high tone

44.1.4 Include File

`/usr/include/celf/mp_cs.h`

44.1.5 Functional Description

This function gets the reconnection tone information to the non-volatile memory.

45.Get Noise Cancel

45.1 Symbol: celf_mp_cs_get_noise_cancel

45.1.1 Syntax

celfMpStatus celf_mp_cs_get_noise_cancel (

void);

45.1.2 Argument

None.

45.1.3 Return Value

Type:	celfMpCsNoiseCancel
I/O:	O

Description:

`celf_mp_cs_get_noise_cancel()` **shall** return one of the values defined:

CELF_CS_ON:	Noise canceller ON
CELF_CS_OFF:	Noise canceller OFF

45.1.4 Include File

`/usr/include/celf/mp_cs.h`

45.1.5 Functional Description

This function gets the noise canceller status.

46.Set Noise Cancel

46.1 Symbol: celf_mp_cs_set_noise_cancel

46.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_noise_cancel (
            celfMpCsNoiseCancel      mode);
```

46.1.2 Argument

Name:	mode
Type:	celfMpCsNoiseCancel
I/O:	I

Description:

Reconnection tone to be set

CELF_CS_ON:	Noise canceller ON
CELF_CS_OFF:	Noise canceller OFF

46.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_noise_cancel()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

46.1.4 Include File

`/usr/include/celf/mp_cs.h`

46.1.5 Functional Description

This function sets the noise canceller off or on.

47.Get Quality Alarm

47.1 Symbol: celf_mp_cs_get_quality_alarm

47.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_quality_alarm(
        void);
```

47.1.2 Argument

None.

47.1.3 Return Value

Type:	celfMpCsQualAlarm
I/O:	O

Description:

`celf_mp_cs_get_quality_alarm()` **shall** return one of the values defined:

CELF_CS_QUALITY_ALM_OFF:	Quality alarm OFF
CELF_CS_QUALITY_ALM_LOW:	Quality alarm ON low tone
CELF_CS_QUALITY_ALM_HI:	Quality alarm ON high tone

47.1.4 Include File

`/usr/include/celf/mp_cs.h`

47.1.5 Functional Description

This function gets the status of the call quality alarm sound.

48.Set Quality Alarm

48.1 Symbol: celf_mp_cs_set_quality_alarm

48.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_quality_alarm (
        celfMpCsQualAlarm      mode);
```

48.1.2 Argument

Name:	mode
Type:	celfMpCsQualAlarm
I/O:	I

Description:

CELF_CS_QUALITY_ALM_OFF:	Quality alarm OFF
CELF_CS_QUALITY_ALM_LOW:	Quality alarm ON low tone
CELF_CS_QUALITY_ALM_HI:	Quality alarm ON high tone

48.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_quality_alarm()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

48.1.4 Include File

`/usr/include/celf/mp_cs.h`

48.1.5 Functional Description

This function sets the call quality alarm sound.

49.Get Noise Cancel Permit

49.1 Symbol: celf_mp_cs_get_noise_cancel_permit

49.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_noise_cancel_permit(
        void);
```

49.1.2 Argument

None.

49.1.3 Return Value

Type: celfMpCsQualAlarm

I/O: O

Description:

`celf_mp_cs_get_noise_cancel_permit()` **shall** return one of the values defined:

CELF_CS_ON: Noise canceller permission

CELF_CS_OFF: Noise canceller non-permission

49.1.4 Include File

`/usr/include/celf/mp_cs.h`

49.1.5 Functional Description

This function obtains whether noise canceller is permitted or not.

50.Set High Priority communication mode

50.1 Symbol: celf_mp_cs_set_hi_prio_com

50.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_hi_prio_com (
        celfMpCsHiPrioCom       mode);
```

50.1.2 Argument

Name:	mode
Type:	celfMpCsHiPrioCom
I/O:	I

Description:

Reconnection tone to be set

CELF_CS_COMPRI_NONE:	No setting
CELF_CS_COMPRI_VOICE:	Voice
CELF_CS_COMPRI_PACKET:	Packet

50.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_hi_prio_com()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

50.1.4 Include File

`/usr/include/celf/mp_cs.h`

50.1.5 Functional Description

This function sets the high priority communication mode either on the voice communication or on the packet communication.

51.Get Phone Answering Sound Activation

51.1 Symbol: celf_mp_cs_get_vm_sound_status

51.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_vm_sound_status(
        void);
```

51.1.2 Argument

None.

51.1.3 Return Value

Type:	celfMpCsVmSound
I/O:	O

Description:

`celf_mp_cs_get_vm_sound_status()` **shall** return one of the values defined:

CELF_CS_ON: Message sound ON

CELF_CS_OFF: Message sound OFF

51.1.4 Include File

`/usr/include/celf/mp_cs.h`

51.1.5 Functional Description

This function gets the setting status.

IF the setting status is ON, the phone sounds, when the number of phone-answering message is increased.

52.Set Phone Answering Sound Activation

52.1 Symbol: celf_mp_cs_set_vm_sound_status

52.1.1 Syntax

celfMpStatus celf_mp_cs_get_vm_sound_status (

celfMpCsQualAlarm mode);

52.1.2 Argument

Type: celfMpCsVmSound

I/O: O

Description:

CELF_CS_ON: Message sound ON

CELF_CS_OFF: Message sound OFF

52.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_vm_sound_status()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

52.1.4 Include File

`/usr/include/celf/mp_cs.h`

52.1.5 Functional Description

This function sets the phone sounds status whether the phone sounds or not.

53.Get Automatic Receive Status

53.1 Symbol: celf_mp_cs_get_auto_rcv_status

53.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_auto_rcv_status(
        void);
```

53.1.2 Argument

None.

53.1.3 Return Value

Type: celfMpCsVmSound

I/O: O

Description:

`celf_mp_cs_get_auto_rcv_status()` **shall** return one of the values defined:

CELF_CS_ON: Automatic incoming call ON

CELF_CS_OFF: Automatic incoming call OFF

53.1.4 Include File

`/usr/include/celf/mp_cs.h`

53.1.5 Functional Description

This function obtains the status of automatic incoming call.

The status is ON or OFF.

54.Set Automatic Receive Status

54.1 Symbol: celf_mp_cs_set_auto_rcv_status

54.1.1 Syntax

celfMpStatus celf_mp_cs_set_auto_rcv_status (

celfMpCsAutoRcv mode);

54.1.2 Argument

Type:	celfMpCsAutoRcv
I/O:	O

Description:

CELF_CS_ON: Automatic incoming call ON

CELF_CS_OFF: Automatic incoming call OFF

54.1.3 Return Value

Type:	celfMpStatus
I/O:	O

Description:

`celf_mp_cs_set_auto_rcv_status()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

54.1.4 Include File

`/usr/include/celf/mp_cs.h`

54.1.5 Functional Description

This function sets the automatic incoming call status.

55.Get Automatic Timer

55.1 Symbol: celf_mp_cs_get_auto_timer

55.1.1 Syntax

celfMpStatus celf_mp_cs_get_auto_timer(

void);

55.1.2 Argument

None.

55.1.3 Return Value

Type: celfMpCsTimer

I/O: O

Description:

`celf_mp_cs_get_auto_timer()` **shall** return one of the values defined:

1 to 120 seconds

55.1.4 Include File

`/usr/include/celf/mp_cs.h`

55.1.5 Functional Description

This function obtains the timer value of the automatic incoming call.

The timer value is the duration of sounding of the ring alert.

56.Set Automatic Timer

56.1 Symbol: celf_mp_cs_set_auto_timer

56.1.1 Syntax

celfMpStatus celf_mp_cs_set_auto_timer (

celfMpCsTimer time);

56.1.2 Argument

Type: celfMpCsTimer

I/O: O

Description:

1 to 120 seconds

56.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_auto_timer()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

56.1.4 Include File

`/usr/include/celf/mp_cs.h`

56.1.5 Functional Description

This function sets the timer value of the automatic incoming call.

57.Get Reset Date

57.1 Symbol: celf_mp_cs_get_reset_date

57.1.1 Syntax

celfMpStatus celf_mp_cs_get_reset_date(

    celfMpCsDate * reset date);

57.1.2 Argument

Type: celfMpCsDate

I/O: O

Description:

Accumulated date record

See section 0.1 for details.

57.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_get_reset_date()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

57.1.4 Include File

`/usr/include/celf/mp_cs.h`

57.1.5 Functional Description

This function obtains the date and time that the accumulated date record was reset.

The value is obtained from non-volatile memory.

58.Set Reset Date

58.1 Symbol: celf_mp_cs_set_reset_date

58.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_reset_date(
            void);
```

58.1.2 Argument

None.

58.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_reset_date()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

58.1.4 Include File

`/usr/include/celf/mp_cs.h`

58.1.5 Functional Description

This function sets the current date and time as the reset date and time of the accumulated date record.

The value set to non-volatile memory.

59.Get Call Start Time

59.1 Symbol: celf_mp_cs_get_call_start_time

59.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_call_start_time(
        void );
```

59.1.2 Argument

None.

59.1.3 Return Value

Type:	celfMpTime
I/O:	O

Description:

`celf_mp_cs_get_call_start_time()` **shall** return one of the values defined:

0 to 99 seconds

59.1.4 Include File

`/usr/include/celf/mp_cs.h`

59.1.5 Functional Description

This function gets the duration between the arrival of incoming call and the start of sounding of the ring alert. This duration is called the silent time.

This function is effective that the number of this incoming call is unregistered with the phone book.

60.Set Call Start Time

60.1 Symbol: celf_mp_cs_set_call_start_time

60.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_call_start_time(
            celfMpCsTimer   time);
```

60.1.2 Argument

Type: celfMpCsTimer

I/O: O

Description:

1 to 99 seconds

60.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_call_start_time()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

60.1.4 Include File

`/usr/include/celf/mp_cs.h`

60.1.5 Functional Description

This function sets the silent time.

Refer to get calling operation start time.

61.Get Call Recorded

61.1 Symbol: celf_mp_cs_get_call_recorded

61.1.1 Syntax

```
celfMpStatus celf_mp_cs_get_call_recorded(
        void );
```

61.1.2 Argument

None.

61.1.3 Return Value

Type:	celfMpSetting
I/O:	O

Description:

`celf_mp_cs_get_call_recorded()` **shall** return one of the values defined:

CELF_CS_ON:	Setting ON
CELF_CS_OFF:	Setting OFF

61.1.4 Include File

`/usr/include/celf/mp_cs.h`

61.1.5 Functional Description

This function gets the setting condition of whether the silent call is recorded in the absent incoming call log, or not.

The absent incoming call log is the log that records no-responded incoming call.

The silent call is the incoming call, which disconnects within the silent time.

Refer to “Get calling operation start time”.

62.Set Call Recorded

62.1 Symbol: celf_mp_cs_set_call_recorded

62.1.1 Syntax

```
celfMpStatus celf_mp_cs_set_call_recorded(
        celfMpCsSetting  mode);
```

62.1.2 Argument

Type: celfMpCsSetting

I/O: O

Description:

CELF_CS_ON: Setting ON

CELF_CS_OFF: Setting OFF

62.1.3 Return Value

Type: celfMpStatus

I/O: O

Description:

`celf_mp_cs_set_call_start_time()` **shall** return one of the values defined:

CELF_MP _STATUS_OK:	successful completion
CELF_MP_STATUS_ERR:	Other unsuccessful completion.

62.1.4 Include File

`/usr/include/celf/mp_cs.h`

62.1.5 Functional Description

This function sets the setting condition of whether the silent call is recorded in the absent incoming call log, or not.

Refer to “Get recording condition to absent incoming call log”.