



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Linux based 3G Specification

Multimedia Mobile Phone API

Circuit Switched Communication Service

Document: CELF_MPP_CS_FR1_20060301

WARNING: This is a working draft for review only, it is NOT a published specification of the CE Linux Forum. It is likely that further substantial changes will be made in the course of review and issue resolution. Send comments on this version to:

MppApiComments@tree.celinuxforum.org

DRAFT

28 **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
2.2.1a	NEC comments	NEC	AK	05/11/08
2.2.2	Review comments	Sharp	AK	05/11/20
2.2.2a	Template Change		AK	05/11/21
2.2.3	AppId ref removed Extract Common		AK	05/12/28
FR1	Version with line numbers for Formal Review		Scott Preece	06/3/1

DRAFT

29

30	0. Introduction.....	15
31	References.....	16
32	0.1.1 Normative	16
33	0.1.2 Informative	16
34	1. Primitives.....	17
35	1.1 Constants.....	17
36	1.1.1 Line type	17
37	1.1.2 Dial Number	17
38	1.1.3 TAF address.....	17
39	1.2 Enums.....	18
40	1.2.1 Voice communication status (CelfMpCsComStatus)	18
41	1.2.2 Forwarding result (CELF_CS_FW_RESULT).....	19
42	1.2.3 Forwarding result details (*Set only at forwarding failure.).....	19
43	1.2.4 Communication type (CELF_CS_BTTYPE).....	19
44	1.2.5 Call Reference Status.....	19
45	1.2.6 Call Status.....	19
46	1.2.7 Existence of continuation data.....	20
47	1.2.8 BT sound flag	20
48	1.2.9 Cause of NoCLI.....	20
49	1.2.10 Dial number display identifier.....	20
50	1.2.11 Redirect number display identifier.....	20
51	1.2.12 Signal information	21
52	1.2.13 Originating Number notification (CELF_NOTICE).....	21
53	1.2.14 Line status.....	21
54	1.2.15 Normal and emergency originating restriction	21
55	1.2.16 Receive level.....	21
56	1.2.17 Area status information.....	22
57	1.2.18 RRC mode	22
58	1.2.19 Service status	22
59	1.2.20 Restriction status.....	22
60	1.2.21 Identifying flag	22
61	1.3 Data Types and Structures	23
62	1.3.1 Circuit switched status notification event structure	23
63	1.3.2 Call duration notification event structure.....	23
64	1.3.3 Disconnection cause notification event structure.....	23
65	1.3.4 Disconnection cause information structure.....	23
66	1.3.5 Forwarding result notification event structure	24
67	1.3.6 Forwarding result structure.....	24
68	1.3.7 Off-hook transmission timeout event structure.....	24
69	1.3.8 Connection Destination Information.....	24
70	1.3.9 Connection Request (CELF_CON_REQ)	25
71	1.3.10 Redirection number.....	25
72	1.3.11 Channel Number Information	25
73	1.3.12 DCF Event Structure.....	25
74	1.3.13 Line status change notification event structure	26
75	1.3.14 Restriction display information structure.....	26
76	1.3.15 Receive level change notification event structure.....	26
77	1.3.16 Line Status structure	26
78	1.3.17 Supplementary service data structure	27
79	1.3.18 Response Message Data Structure	27

80	1.3.19	Date Format Structure.....	27
81	1.4	Events Type.....	28
82	1.4.1	DCF Event Type	28
83	1.4.2	CCP Notification type.....	28
84	1.4.3	Notification type	29
85	1.4.4	Restriction status.....	29
86	2.	Start Notification.....	30
87	2.1	Symbol: celf_mp_cs_notification_start.....	30
88	2.1.1	Syntax	30
89	2.1.2	Argument	30
90	2.1.3	Return Value.....	31
91	2.1.4	Include File	31
92	2.1.5	Functional Description.....	31
93	3.	Stop Notification	32
94	3.1	Symbol: celf_mp_cs_notification_stop	32
95	3.1.1	Syntax	32
96	3.1.2	Argument	32
97	3.1.3	Return Value.....	32
98	3.1.4	Include File	33
99	3.1.5	Functional Description.....	33
100	4.	Get Voice Communication Status	34
101	4.1	Symbol: celf_mp_cs_get_com_status.....	34
102	4.1.1	Syntax	34
103	4.1.2	Argument	34
104	4.1.3	Return Value.....	34
105	4.1.4	Include File	34
106	4.1.5	Functional Description.....	34
107	5.	Get Connection Information to Other Party.....	35
108	5.1	Symbol: celf_mp_cs_get_con_info_ref	35
109	5.1.1	Syntax	35
110	5.1.2	Argument	35
111	5.1.3	Return Value.....	35
112	5.1.4	Include File	36
113	5.1.5	Functional Description.....	36
114	6.	Get Call Duration.....	37
115	6.1	Symbol: celf_mp_cs_get_call_duration	37
116	6.1.1	Syntax	37
117	6.1.2	Argument	37
118	6.1.3	Return Value.....	37
119	6.1.4	Include File	37
120	6.1.5	Functional Description.....	37
121	7.	Off-Hook Notification.....	38
122	7.1	Symbol: celf_mp_cs_notification_off_hook	38
123	7.1.1	Syntax	38
124	7.1.2	Argument	38
125	7.1.3	Return Value.....	38
126	7.1.4	Include File	39

127	7.1.5	Functional Description.....	39
128	8.	<i>Disconnect</i>	40
129	8.1	Symbol: celf_mp_cs_disconnect	40
130	8.1.1	Syntax	40
131	8.1.2	Argument.....	40
132	8.1.3	Return Value.....	40
133	8.1.4	Include File	40
134	8.1.5	Functional Description.....	40
135	9.	<i>Dial</i>	42
136	9.1	Symbol: celf_mp_cs_dial	42
137	9.1.1	Syntax	42
138	9.1.2	Argument.....	42
139	9.1.3	Return Value.....	42
140	9.1.4	Include File	43
141	9.1.5	Functional Description.....	43
142	10.	<i>Dial Complete</i>	44
143	10.1	Symbol: celf_mp_cs_dial_end	44
144	10.1.1	Syntax	44
145	10.1.2	Argument.....	44
146	10.1.3	Return Value.....	44
147	10.1.4	Include File	44
148	10.1.5	Functional Description.....	44
149	11.	<i>Response to Incoming Call</i>	46
150	11.1	Symbol: celf_mp_cs_call_rev	46
151	11.1.1	Syntax	46
152	11.1.2	Argument.....	46
153	11.1.3	Return Value.....	46
154	11.1.4	Include File.....	46
155	11.1.5	Functional Description.....	46
156	12.	<i>Forward Incoming Call</i>	48
157	12.1	Symbol: celf_mp_cs_call_forward	48
158	12.1.1	Syntax	48
159	12.1.2	Argument.....	48
160	12.1.3	Return Value.....	48
161	12.1.4	Include File	48
162	12.1.5	Functional Description.....	48
163	13.	<i>Forward to Voice Mail System</i>	50
164	13.1	Symbol: celf_mp_cs_call_forward_voice_msg	50
165	13.1.1	Syntax	50
166	13.1.2	Argument.....	50
167	13.1.3	Return Value.....	50
168	13.1.4	Include File	50
169	13.1.5	Functional Description.....	50
170	14.	<i>Call Hold</i>	52
171	14.1	Symbol: celf_mp_cs_call_hold	52
172	14.1.1	Syntax	52

173	14.1.2	Argument	52
174	14.1.3	Return Value	52
175	14.1.4	Include File	52
176	14.1.5	Functional Description	52
177	15.	<i>Call Reject</i>	54
178	15.1	Symbol: <i>celf_mp_cs_call_reject</i>	54
179	15.1.1	Syntax	54
180	15.1.2	Argument	54
181	15.1.3	Return Value	54
182	15.1.4	Include File	54
183	15.1.5	Functional Description	54
184	16.	<i>Multi Party Call</i>	56
185	16.1	Symbol: <i>celf_mp_cs_mp_call</i>	56
186	16.1.1	Syntax	56
187	16.1.2	Argument	56
188	16.1.3	Return Value	56
189	16.1.4	Include File	57
190	16.1.5	Functional Description	57
191	17.	<i>On-Hook Originating</i>	59
192	17.1	Symbol: <i>celf_mp_cs_originating_on_hook</i>	59
193	17.1.1	Syntax	59
194	17.1.2	Argument	59
195	17.1.3	Return Value	59
196	17.1.4	Include File	59
197	17.1.5	Functional Description	60
198	18.	<i>Get Call Reference</i>	61
199	18.1	Symbol: <i>celf_mp_cs_get_call_reference</i>	61
200	18.1.1	Syntax	61
201	18.1.2	Argument	61
202	18.1.3	Return Value	61
203	18.1.4	Include File	61
204	18.1.5	Functional Description	61
205	19.	<i>Start DCF message notification</i>	63
206	19.1	Symbol: <i>celf_mp_cs_DCF_notification_start</i>	63
207	19.1.1	Syntax	63
208	19.1.2	Argument	63
209	19.1.3	Return Value	64
210	19.1.4	Include File	64
211	19.1.5	Functional Description	64
212	20.	<i>Stop DCF message notification</i>	66
213	20.1	Symbol: <i>celf_mp_cs_DCF_notification_stop</i>	66
214	20.1.1	Syntax	66
215	20.1.2	Argument	66
216	20.1.3	Return Value	66
217	20.1.4	Include File	67
218	20.1.5	Functional Description	67
219	21.	<i>Voice Message Notification</i>	68

220	21.1 Symbol: celf_mp_cs_voice_msg_notify	68
221	21.1.1 Syntax	68
222	21.1.2 Argument	68
223	21.1.3 Return Value	68
224	21.1.4 Include File	68
225	21.1.5 Functional Description.....	68
226	22. Hold Tone Start	69
227	22.1 Symbol: celf_mp_cs_hold_tone_start	69
228	22.1.1 Syntax	69
229	22.1.2 Argument	69
230	22.1.3 Return Value	69
231	22.1.4 Include File	69
232	22.1.5 Functional Description.....	69
233	23. Hold Tone Stop	70
234	23.1 Symbol: celf_mp_cs_hold_tone_stop	70
235	23.1.1 Syntax	70
236	23.1.2 Argument	70
237	23.1.3 Return Value	70
238	23.1.4 Include File	70
239	23.1.5 Functional Description.....	70
240	24. Get 64K / AV Communication Status	71
241	24.1 Symbol: celf_mp_cs_get_UD_com_stat	71
242	24.1.1 Syntax	71
243	24.1.2 Argument	71
244	24.1.3 Return Value	71
245	24.1.4 Include File	71
246	24.1.5 Functional Description.....	71
247	25. Get internal/external AV Communication Status	72
248	25.1 Symbol: celf_mp_cs_get_AV_com_stat	72
249	25.1.1 Syntax	72
250	25.1.2 Argument	72
251	25.1.3 Return Value	72
252	25.1.4 Include File	72
253	25.1.5 Functional Description.....	72
254	26. Get Communication Status	73
255	26.1 Symbol: celf_mp_cs_get_com_stat	73
256	26.1.1 Syntax	73
257	26.1.2 Argument	73
258	26.1.3 Return Value	73
259	26.1.4 Include File	74
260	26.1.5 Functional Description.....	74
261	27. Start Line Status Notification	75
262	27.1 Symbol: celf_mp_cs_line_status_notification_start	75
263	27.1.1 Syntax	75
264	27.1.2 Argument	75
265	27.1.3 Return Value	75
266	27.1.4 Include File	76

267	27.1.5	Functional Description.....	76
268	28.	<i>Stop Line Status Notification</i>	77
269	28.1	Symbol: <code>celf_mp_cs_line_status_notification_stop</code>	77
270	28.1.1	Syntax	77
271	28.1.2	Argument	77
272	28.1.3	Return Value.....	77
273	28.1.4	Include File	78
274	28.1.5	Functional Description.....	78
275	29.	<i>Get Reception Level</i>	79
276	29.1	Symbol: <code>celf_mp_cs_get_reception_level</code>	79
277	29.1.1	Syntax	79
278	29.1.2	Argument	79
279	29.1.3	Return Value.....	79
280	29.1.4	Include File	79
281	29.1.5	Functional Description.....	79
282	30.	<i>Get Line Status</i>	80
283	30.1	Symbol: <code>celf_mp_cs_get_line_status</code>	80
284	30.1.1	Syntax	80
285	30.1.2	Argument	80
286	30.1.3	Return Value.....	80
287	30.1.4	Include File	80
288	30.1.5	Functional Description.....	80
289	31.	<i>Get Coverage Status</i>	81
290	31.1	Symbol: <code>celf_mp_cs_get_coverage_status</code>	81
291	31.1.1	Syntax	81
292	31.1.2	Argument	81
293	31.1.3	Return Value.....	81
294	31.1.4	Include File	81
295	31.1.5	Functional Description.....	81
296	32.	<i>Get Voice Mail Information</i>	83
297	32.1	Symbol: <code>celf_mp_cs_get_vm_info</code>	83
298	32.1.1	Syntax	83
299	32.1.2	Argument	83
300	32.1.3	Return Value.....	83
301	32.1.4	Include File	83
302	32.1.5	Functional Description.....	83
303	33.	<i>Set Voice Mail Information</i>	84
304	33.1	Symbol: <code>celf_mp_cs_set_vm_info</code>	84
305	33.1.1	Syntax	84
306	33.1.2	Argument	84
307	33.1.3	Return Value.....	84
308	33.1.4	Include File	84
309	33.1.5	Functional Description.....	84
310	34.	<i>Get Call Selection</i>	85
311	34.1	Symbol: <code>celf_mp_cs_get_call_select</code>	85
312	34.1.1	Syntax	85

313	34.1.2	Argument	85
314	34.1.3	Return Value	85
315	34.1.4	Include File	85
316	34.1.5	Functional Description	85
317	35.	<i>Set Call Selection</i>	86
318	35.1	Symbol: <code>celf_mp_cs_set_call_select</code>	86
319	35.1.1	Syntax	86
320	35.1.2	Argument	86
321	35.1.3	Return Value	86
322	35.1.4	Include File	86
323	35.1.5	Functional Description	86
324	36.	<i>Set Service Information</i>	87
325	36.1	Symbol: <code>celf_mp_cs_set_service_info</code>	87
326	36.1.1	Syntax	87
327	36.1.2	Argument	87
328	36.1.3	Return Value	87
329	36.1.4	Include File	87
330	36.1.5	Functional Description	87
331	37.	<i>Get Service Information</i>	89
332	37.1	Symbol: <code>celf_mp_cs_get_service_info</code>	89
333	37.1.1	Syntax	89
334	37.1.2	Argument	89
335	37.1.3	Return Value	89
336	37.1.4	Include File	89
337	37.1.5	Functional Description	89
338	38.	<i>Delete Service Information</i>	90
339	38.1	Symbol: <code>celf_mp_cs_del_service_info</code>	90
340	38.1.1	Syntax	90
341	38.1.2	Argument	90
342	38.1.3	Return Value	90
343	38.1.4	Include File	90
344	38.1.5	Functional Description	90
345	39.	<i>Remove Service Information</i>	91
346	39.1	Symbol: <code>celf_mp_cs_remove_all_service_info</code>	91
347	39.1.1	Syntax	91
348	39.1.2	Argument	91
349	39.1.3	Return Value	91
350	39.1.4	Include File	91
351	39.1.5	Functional Description	91
352	40.	<i>Set Response Message Settings</i>	92
353	40.1	Symbol: <code>celf_mp_cs_set_resp_msg</code>	92
354	40.1.1	Syntax	92
355	40.1.2	Argument	92
356	40.1.3	Return Value	92
357	40.1.4	Include File	92
358	40.1.5	Functional Description	92
359	41.	<i>Get Response Message Settings</i>	94

360	41.1	Symbol: celf_mp_cs_get_resp_msg	94
361	41.1.1	Syntax	94
362	41.1.2	Argument	94
363	41.1.3	Return Value	94
364	41.1.4	Include File	94
365	41.1.5	Functional Description.....	94
366	42.	Delete Response Message Settings	95
367	42.1	Symbol: celf_mp_cs_del_resp_msg	95
368	42.1.1	Syntax	95
369	42.1.2	Argument	95
370	42.1.3	Return Value	95
371	42.1.4	Include File	95
372	42.1.5	Functional Description.....	95
373	43.	Remove All Response Message Settings	96
374	43.1	Symbol: celf_mp_cs_remove_all_resp_msg	96
375	43.1.1	Syntax	96
376	43.1.2	Argument	96
377	43.1.3	Return Value	96
378	43.1.4	Include File	96
379	43.1.5	Functional Description.....	96
380	44.	Set Reconnection Tone	97
381	44.1	Symbol: celf_mp_cs_set_reconnection_tone	97
382	44.1.1	Syntax	97
383	44.1.2	Argument	97
384	44.1.3	Return Value	97
385	44.1.4	Include File	97
386	44.1.5	Functional Description.....	97
387	45.	Get Reconnection Tone	98
388	45.1	Symbol: celf_mp_cs_get_reconnection_tone	98
389	45.1.1	Syntax	98
390	45.1.2	Argument	98
391	45.1.3	Return Value	98
392	45.1.4	Include File	98
393	45.1.5	Functional Description.....	98
394	46.	Get Noise Cancel	99
395	46.1	Symbol: celf_mp_cs_get_noise_cancel	99
396	46.1.1	Syntax	99
397	46.1.2	Argument	99
398	46.1.3	Return Value	99
399	46.1.4	Include File	99
400	46.1.5	Functional Description.....	99
401	47.	Set Noise Cancel	100
402	47.1	Symbol: celf_mp_cs_set_noise_cancel	100
403	47.1.1	Syntax	100
404	47.1.2	Argument	100
405	47.1.3	Return Value	100
406	47.1.4	Include File	100

407	47.1.5	Functional Description.....	100
408	48.	<i>Get Quality Alarm</i>	101
409	48.1	Symbol: celf_mp_cs_get_quality_alarm	101
410	48.1.1	Syntax	101
411	48.1.2	Argument	101
412	48.1.3	Return Value.....	101
413	48.1.4	Include File	101
414	48.1.5	Functional Description.....	101
415	49.	<i>Set Quality Alarm</i>	102
416	49.1	Symbol: celf_mp_cs_set_quality_alarm	102
417	49.1.1	Syntax	102
418	49.1.2	Argument	102
419	49.1.3	Return Value.....	102
420	49.1.4	Include File	102
421	49.1.5	Functional Description.....	102
422	50.	<i>Get Noise Cancel Permit</i>	103
423	50.1	Symbol: celf_mp_cs_get_noise_cancel_permit	103
424	50.1.1	Syntax	103
425	50.1.2	Argument	103
426	50.1.3	Return Value.....	103
427	50.1.4	Include File	103
428	50.1.5	Functional Description.....	103
429	51.	<i>Set High Priority communication mode</i>	104
430	51.1	Symbol: celf_mp_cs_set_hi_prio_com	104
431	51.1.1	Syntax	104
432	51.1.2	Argument	104
433	51.1.3	Return Value.....	104
434	51.1.4	Include File	104
435	51.1.5	Functional Description.....	104
436	52.	<i>Get Phone Answering Sound Activation</i>	105
437	52.1	Symbol: celf_mp_cs_get_vm_sound_status	105
438	52.1.1	Syntax	105
439	52.1.2	Argument	105
440	52.1.3	Return Value.....	105
441	52.1.4	Include File	105
442	52.1.5	Functional Description.....	105
443	53.	<i>Set Phone Answering Sound Activation</i>	106
444	53.1	Symbol: celf_mp_cs_set_vm_sound_status	106
445	53.1.1	Syntax	106
446	53.1.2	Argument	106
447	53.1.3	Return Value.....	106
448	53.1.4	Include File	106
449	53.1.5	Functional Description.....	106
450	54.	<i>Get Automatic Receive Status</i>	107
451	54.1	Symbol: celf_mp_cs_get_auto_rcv_status	107
452	54.1.1	Syntax	107

453	54.1.2	Argument	107
454	54.1.3	Return Value	107
455	54.1.4	Include File	107
456	54.1.5	Functional Description.....	107
457	55.	<i>Set Automatic Receive Status</i>	108
458	55.1	Symbol: <i>celf_mp_cs_set_auto_rcv_status</i>	108
459	55.1.1	Syntax	108
460	55.1.2	Argument	108
461	55.1.3	Return Value	108
462	55.1.4	Include File	108
463	55.1.5	Functional Description.....	108
464	56.	<i>Get Automatic Timer</i>	109
465	56.1	Symbol: <i>celf_mp_cs_get_auto_timer</i>	109
466	56.1.1	Syntax	109
467	56.1.2	Argument	109
468	56.1.3	Return Value	109
469	56.1.4	Include File	109
470	56.1.5	Functional Description.....	109
471	57.	<i>Set Automatic Timer</i>	110
472	57.1	Symbol: <i>celf_mp_cs_set_auto_timer</i>	110
473	57.1.1	Syntax	110
474	57.1.2	Argument	110
475	57.1.3	Return Value	110
476	57.1.4	Include File	110
477	57.1.5	Functional Description.....	110
478	58.	<i>Get Reset Date</i>	111
479	58.1	Symbol: <i>celf_mp_cs_get_reset_date</i>	111
480	58.1.1	Syntax	111
481	58.1.2	Argument	111
482	58.1.3	Return Value	111
483	58.1.4	Include File	111
484	58.1.5	Functional Description.....	111
485	59.	<i>Set Reset Date</i>	112
486	59.1	Symbol: <i>celf_mp_cs_set_reset_date</i>	112
487	59.1.1	Syntax	112
488	59.1.2	Argument	112
489	59.1.3	Return Value	112
490	59.1.4	Include File	112
491	59.1.5	Functional Description.....	112
492	60.	<i>Get Call Start Time</i>	113
493	60.1	Symbol: <i>celf_mp_cs_get_call_start_time</i>	113
494	60.1.1	Syntax	113
495	60.1.2	Argument	113
496	60.1.3	Return Value	113
497	60.1.4	Include File	113
498	60.1.5	Functional Description.....	113
499	61.	<i>Set Call Start Time</i>	114

500 **61.1 Symbol: celf_mp_cs_set_call_start_time..... 114**
501 61.1.1 Syntax 114
502 61.1.2 Argument 114
503 61.1.3 Return Value 114
504 61.1.4 Include File 114
505 61.1.5 Functional Description..... 114

506 **62. *Get Call Recorded* 115**

507 **62.1 Symbol: celf_mp_cs_get_call_recorded..... 115**
508 62.1.1 Syntax 115
509 62.1.2 Argument 115
510 62.1.3 Return Value 115
511 62.1.4 Include File 115
512 62.1.5 Functional Description..... 115

513 **63. *Set Call Recorded*..... 116**

514 **63.1 Symbol: celf_mp_cs_set_call_recorded 116**
515 63.1.1 Syntax 116
516 63.1.2 Argument 116
517 63.1.3 Return Value 116
518 63.1.4 Include File 116
519 63.1.5 Functional Description..... 116

520

DRAFT

521

0. Introduction

522
523

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

524

Circuit Switched Communication Service includes

525
526
527

- Voice communication service,
- Video communication service, and
- Unrestricted Digital data Communication service.

DRAFT

528

529 **References**

530 **0.1.1 Normative**

531 RFC 2119: “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997,

532 [URL:http://www.ietf.org/rfc/rfc2119.txt](http://www.ietf.org/rfc/rfc2119.txt)

533 RFC 2234: “Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November

534 1997, [URL:http://www.ietf.org/rfc/rfc2234.txt](http://www.ietf.org/rfc/rfc2234.txt)

535

536 **0.1.2 Informative**

DRAFT

537

1. Primitives

538

This section contains the definitions of the data types and constants used in the interfaces of this service.

539

1.1 Constants

540

1.1.1 Line type

541

CELF_CS_LINE_WCDMA WCDMA

542

1.1.2 Dial Number

543

Dial number of the other party

544

This data is valid when this mobile phone originates a call.

545

CELF_CS_DIAL_MAX is 45.

546

1.1.3 TAF address

547

Internal/External TAF type

548

32 to 63: Internal TAF

549

64 to 79: External TAF

550

DRAFT

551 **1.2 Enums**552 **1.2.1 Voice communication status (CelfMpCsComStatus)**

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

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

555 **1.2.1.1 Condition: only one call**

556	CELF_CS_COM_STATUS_WAIT:	Standby
557	CELF_CS_COM_STATUS_RCV:	Under incoming
558	CELF_CS_COM_STATUS_TRN:	Under outgoing
559	CELF_CS_COM_STATUS_DLV:	Under calling
560	CELF_CS_COM_STATUS_TLK:	Under conversation
561	CELF_CS_COM_STATUS_HLD:	Under response hold

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

564	CELF_CS_COM_STATUS_RLS:	Under release
-----	-------------------------	---------------

565 **1.2.1.2 Condition: two call**

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

567	CELF_CS_COM_STATUS_TLK_RCV:	Under conversation and incoming
568	CELF_CS_COM_STATUS_TLK_TRN:	Under conversation and outgoing
569	CELF_CS_COM_STATUS_TLK_DLV:	Under conversation and calling
570	CELF_CS_COM_STATUS_TLK_RSV:	Under conversation and hold
571	CELF_CS_COM_STATUS_TLK_RLS:	Under conversation and release

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

574	CELF_CS_COM_STATUS_TLK_RSV_RCV:	Under conversation, hold, and incoming
-----	---------------------------------	--

575 **1.2.1.3 Condition: only one AV call**

576	CELF_CS_COM_STATUS_RCV_AV:	Under incoming of an AV call
577	CELF_CS_COM_STATUS_TRN_AV:	Under outgoing of an AV call
578	CELF_CS_COM_STATUS_DLV_AV:	Under calling of an AV call
579	CELF_CS_COM_STATUS_TLK_AV:	Under conversation of an AV call
580	CELF_CS_COM_STATUS_HLD_AV:	Under response hold of an AV call
581	CELF_CS_COM_STATUS_RLS_AV:	Under release of an AV call

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

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

584 (b) that two call are both in conversation,

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

586

587 **1.2.2 Forwarding result (CELF_CS_FW_RESULT)**

588 CELF_CS_OK Successful forwarding

589 CELF_CS_ERR Forwarding failure

590

591 **1.2.3 Forwarding result details (*Set only at forwarding failure.)**

592 CELF_CS_FW_ERROR_NO_JOIN Service is not contracted.

593 CELF_CS_FW_ERROR_NO_SETDATA The forwarded destination is not registered.

594 CELF_CS_FW_ERROR_ETC Others

595

596 **1.2.4 Communication type (CELF_CS_BTYPE)**

597 CELF_CS_BTYPE_CS_NONE None (unfixed)

598 CELF_CS_BTYPE_CS_ANY Not Specified

599 CELF_CS_BTYPE_CS_VOICE Voice

600 CELF_CS_BTYPE_CS_UD32UD 32K communication

601 CELF_CS_BTYPE_CS_UD64UD 64K communication

602 CELF_CS_BTYPE_CS_AV32AV 32K communication

603 CELF_CS_BTYPE_CS_AV64AV 64K communication

604

605 **1.2.5 Call Reference Status**

606 CELF_CS_USED: "CN_No" is valid.

607 CELF_CS_UNUSED: "CN_No" is not valid.

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

610

611 **1.2.6 Call Status**

612 Call status for this mobile phone

613 CELF_CS_CHAN_KIND_NULL: Vacant

614 CELF_CS_CHAN_KIND_OFF: Off-hook

615 CELF_CS_CHAN_KIND_TRN: Outgoing call

616 CELF_CS_CHAN_KIND_DLV: Calling

617 CELF_CS_CHAN_KIND_RCV: Incoming call

618 CELF_CS_CHAN_KIND_REQ_T: Response (conversation)

619 (The status of responding mobile phone is conversation.)

620 CELF_CS_CHAN_KIND_ACT: Under conversation

621 CELF_CS_CHAN_KIND_REQ_H: Response (hold)

622 (The status of responding mobile phone is hold.)

623 CELF_CS_CHAN_KIND_HLD: Hold response
624 CELF_CS_CHAN_KIND_RSV: Under hold
625 CELF_CS_CHAN_KIND_REL: Under release
626

627 1.2.7 Existence of continuation data

628 CELF_CS_ON: valid below data
629 CELF_CS_OFF: non valid below data
630 The below data, from "Calling_Dial" to "cause", are valid data if the call status is incoming or conversation
631 and incoming call.
632
633

634 1.2.8 BT sound flag

635 Whether BT sounds in this phone, or not
636 CELF_CS_SOUND_BT_ON: BT tone sounds.
637 CELF_CS_SOUND_BT_OFF: BT tone is being stopped.
638
639

640 1.2.9 Cause of NoCLI

641 The reason why the dial number of other party is not notified.
642 The dial number of other party is in "Calling dial" or "Called dial".
643 CELF_CS_NOCL_NOSRV: service is not supported.
644 CELF_CS_NOCL_USER: user rejects to display.
645 CELF_CS_NOCL_INTRACTSRV: service conflicts.
646 CELF_CS_NOCL_PAYPHON: origination is from a public phone.
647 This data is valid, when next data "num_presentation_indicator", is that Display is impossible.
648

649 1.2.10 Dial number display identifier

650 Whether dial number of other party can be displayed, or not.
651 CELF_CS_PRSENT_IND_ALLOWED: Displayable
652 CELF_CS_PRSENT_IND_RESTRICTED: Impossible to display
653 CELF_CS_PRSENT_IND_NOT_AVAILABLE: Displayable number does not exist.
654 CELF_CS_PRSENT_IND_RESERVE: Reservation
655

656 1.2.11 Redirect number display identifier

657 Whether redirection number can be displayed, or not.
658 CELF_CS_PRSENT_IND_ALLOWED: Displayable

659 CELF_CS_PRSNT_IND_RESTRICTED: Display is impossible.
660 CELF_CS_PRSNT_IND_NOT_AVAILABLE: Displayable number does not exist.
661 CELF_CS_PRSNT_IND_RESERVE: Reservation

662

663 1.2.12 Signal information

664 The type of tone of this phone

665 CELF_CS_SIGNAL_DIAL_TONE_ON: Dial tone on
666 CELF_CS_SIGNAL_RINGBACK_TONE_ON: Ring back tone on
667 CELF_CS_SIGNAL_INTERCEPT_TONE_ON: Intercept tone on
668 CELF_CS_SIGNAL_NW_CONGESTION_TONE_ON: Network congestion tone on
669 CELF_CS_SIGNAL_BUSY_TONE_ON: Busy tone on
670 CELF_CS_SIGNAL_CONFIRM_TONE_ON: Confirm tone on
671 CELF_CS_SIGNAL_ANSWER_TONE_ON: Answer tone on
672 CELF_CS_SIGNAL_CALLWAITING_TONE_ON: Call waiting tone on
673 CELF_CS_SIGNAL_OFFHK_WARNING_TONE_ON: Off-hook warning tone on
674 CELF_CS_SIGNAL_TONES_OFF: Tones off
675 CELF_CS_SIGNAL_ALERTING_OFF: Alerting off
676 CELF_CS_SIGNAL_UNSETTING: Signal information is not set.

677 1.2.13 Originating Number notification (CELF_NOTICE)

678 Whether the originating dial number is notified or not.

679 CELF_CS_NOTICE_ON: Notified
680 CELF_CS_NOTICE_OFF: Not notified
681 CELF_CS_NOTICE_NOSET: No setting

682 1.2.14 Line status

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

685 1.2.15 Normal and emergency originating restriction

686 CELF_CS_LINE_RESTRICT_DATA_ON: With originating restriction
687 CELF_CS_LINE_RESTRICT_DATA_OFF: Without originating restriction

688 1.2.16 Receive level

689 CELF_CS_RSSI_LEVEL_0: Receive level 0
690 CELF_CS_RSSI_LEVEL_1: Receive level 1
691 CELF_CS_RSSI_LEVEL_2: Receive level 2
692 CELF_CS_RSSI_LEVEL_3: Receive level 3

693 **1.2.17 Area status information**

694 CELF_CS_LINE_CVR_STATUS_IN IN
695 CELF_CS_LINE_CVR_STATUS_OUT OUT

696

697 **1.2.18 RRC mode**

698 CELF_CS_LINE_RRC_MODE_IDLE idle-mode
699 CELF_CS_LINE_RRC_MODE_UTRAN utran-connected-mode

700

701 Network identification information

702 CELF_CS_LINE_NETWORK_HOME home
703 CELF_CS_LINE_NETWORK_VISIT visit
704 CELF_CS_LINE_NO_DATA No data

705

706 **1.2.19 Service status**

707 CELF_CS_LINE_SRV_STATUS_CS CS is in service.
708 CELF_CS_LINE_SRV_STATUS_PS PS is in service.
709 CELF_CS_LINE_SRV_STATUS_CSPS CS and PS are in service.
710 CELF_CS_LINE_NO_DATA No data

711 CS is the circuit switched communication service, and
712 PS is the packet switched communication service.

713

714 **1.2.20 Restriction status**

715 CELF_CS_LINE_RESTRICT_ON In traffic restriction
716 CELF_CS_LINE_RESTRICT_OFF Out of traffic restriction

717

718 **1.2.21 Identifying flag**

719 Enum CELF_CS_FLAG {
720 CELF_CS_NO_FLAG, // no Flag
721 CELF_CS_OPT_FLAG, // special number
722 CELF_CS_USSD_FLAG // USSD number
723 }

724

725

726

727 1.3 Data Types and Structures

728 1.3.1 Circuit switched status notification event structure

729 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

730 category = VoiceNotify;

731 subtype = VoiceNotify_ConnInfo;

732 The value of field “info” is from enum CelfMpCsComStatus.

733 The field “data” carries:

```
734         CELF_CS_RES_CHG_INF      res_chg_inf;    // to be used in the case of:  
735                                     // Restriction display information structure
```

737 1.3.2 Call duration notification event structure

738 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

739 category = VoiceNotify;

740 subtype = VoiceNotify_TelCallTime;

741 The value of field “info” is Call duration (seconds).

742 The field “data” is unused.

743

744 1.3.3 Disconnection cause notification event structure

745 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

746 category = VoiceNotify;

747 subtype = VoiceNotify_DiscCause;

748 The value of field “info” is the call reference.

749 The field “data” carries:

```
750         CELF_CS_DISC_CAUSE cme; //Disconnection cause information structure
```

751

752 1.3.4 Disconnection cause information structure

```
753 typedef struct {
```

```
754     unsigned char e_code; //Result code flag
```

```
755     unsigned char code; //Result code
```

```
756     unsigned char e_cause1; //Error reason 1 flag
```

```
757     unsigned char cause1; //Error reason 1 (ccpMtCause)
```

```
758     unsigned char e_cause2; //Error reason 2 flag
```

```
759     unsigned char cause2; //Error reason 2 (Cause)
```

```
760 } CELF_CS_DISC_CAUSE ;
```

761

762 1.3.5 Forwarding result notification event structure

763 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

764 category = VoiceNotify;

765 subtype = VoiceNotify_FW_Result

766 The value of field “info” is the call reference.

767 The value of “subinfo” carries the forwarding result.

768 The field “data” carries:

```
769         CELF_CS_FW_RESULT fw_result; // Forwarding result structure
```

770

771 1.3.6 Forwarding result structure

```
772 typedef struct {
```

```
773     int cause ;           //forwarding result details
```

```
774 } CELF_CS_FW_RESULT;
```

775

776 1.3.7 Off-hook transmission timeout event structure

777 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

778 category = VoiceNotify;

779 subtype = VoiceNotify_OffHk_Trn

780 The value of field “info” is the call reference.

781 The field “data” is unused.

782

783 1.3.8 Connection Destination Information

```
784 typedef struct {
```

```
785     int CN_No;           // Call reference
```

```
786     int CN_status;
```

```
787     int continue_flag;
```

```
788     unsigned char Calling_Dial [CELF_CS_DIAL_MAX+1];
```

```
789     unsigned char Called_Dial [CELF_CS_DIAL_MAX+1];
```

```
790     unsigned char BTsound_inf;
```

```
791     CELF_CS_BTTYPE bc_type;
```

```
792     unsigned char taf_address;
```

```
793     unsigned char cause_of_NoCLI;
```

```
794     unsigned char num_presentation_indicator;
```

```
795     unsigned char redirectnum [CELF_CS_DIAL_MAX+1];
```

```
796     unsigned char redirect_presentation_indicator;
```



```
797 unsigned char signal;  
798 CELF_CS_CME cause; // Disconnection cause information structure  
799 } CELF_CS_CONNECT_INF  
800
```

801 1.3.9 Connection Request (CELF_CON_REQ)

```
802 typedef struct {  
803     CELF_CS_BTYPE     type;  
804     unsigned char *   dial_buf;  
805     int               dial_len;  
806     CELF_CS_NOTICE   notice;  
807     unsigned char *   subaddr_buf;  
808     int               subaddr_len;  
809 } CELF_CON_REQ  
810
```

811 1.3.10 Redirection number

812 Destination number of call transfer.
813 redirectnum [CELF_CS_DIAL_MAX+1]
814

815 1.3.11 Channel Number Information

816 CELF_CS_CHANNUM is used to hold call reference information.
817 If a channel is not used, CELF_CS_CHAN_NOUSE is set as the call reference.
818

```
819 Typedef struct {  
820     int ChanNum_00     // Call reference information 00  
821     int ChanNum_01     // Call reference information 01  
822     int ChanNum_02     // Call reference information 02  
823 } CELF_CS_CHANNUM  
824
```

825 1.3.12 DCF Event Structure

826 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:
827 category = VoiceNotify;
828 subtype = Event type;
829 The value of field “info” is the notification type.
830 The value of field “subinfo” is the bearer type
831 The field “data” carries:

832 DCF message structure corresponding to report types.

833

834 1.3.13 Line status change notification event structure

835 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

836 category = VoiceNotify;

837 subtype = VoiceNotify_AreaInfo;

838 The value of field “info” is the line status.

839 The value of field “subinfo” is the line type.

840 The field “data” is unused.

841 1.3.14 Restriction display information structure

842 typedef struct {

843 unsigned char NcRestriction; //Normal originating restriction

844 unsigned char ServiceStatus; //Service status

845 unsigned char EcRestriction; //Emergency originating restriction

846 } CELF_CS_RES_CHG_INF;

847 1.3.15 Receive level change notification event structure

848 In this sub-section, the associated data structure is CELF_MP_EVENT with the following values:

849 category = VoiceNotify;

850 subtype = VoiceNotify_RssiLevel;

851 The value of field “info” is the receive level.

852 The value of field “subinfo” is the line type.

853 The field “data” is unused.

854 1.3.16 Line Status structure

855 typedef struct {

856 unsigned char LineStatus ; //Line status

857 unsigned char CoverageStatus ; //Area status information

858 unsigned char RRcmode ; //RRC mode

859 unsigned char Network ; //Network identification information

860 unsigned char unused; //unused

861 unsigned char ServiceStatus_AREA ; //Service status

862 unsigned char RestrictStatus ; //Restriction status

863 unsigned char NcRestriction ; //Normal originating restriction

864 unsigned char ServiceStatus_RES ; //Service status

865 unsigned char EcRestriction ; //Emergency originating restriction

866 } CELF_CS_AREAREF_CHG_INF ;

867

868 1.3.17 Supplementary service data structure

```
869 typedef struct {
870     CELF_CS_FLAG flag ;
871     char title[CELF_SRVINFO_TITLE];    // Supplementary service name CELF_SRVINFO_TITLE=21
872     char send_no[CELF_SRVINFO_DATA]; // Dial data for accessing the service
873                                     // CELF_SRVINFO_DATA=40
874 } CELF_CS_ADDSRV_DATA;
875
```

876 1.3.18 Response Message Data Structure

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

879

```
880 typedef struct {
881     unsigned char title[CELF_RESMSG_TITLE] ; //Service name
882     unsigned char res_msg[CELF_RESMSG_DATA]; //Dial data
883 } CELF_CS_RESPONSE_MSG_DATA;
884
```

885 1.3.19 Date Format Structure

```
886 typedef struct {
887     unsigned char Month
888     unsigned char Day
889     unsigned char Hour
890     unsigned char Minute
891 } CELF_MP_CS_DATE
892
```

893 1.4 Events Type

894 1.4.1 DCF Event Type

895	VoiceNotify_DCF_Dis	Display-related message
896	VoiceNotify_DCF_History	History-related message
897	VoiceNotify_DCF_Tone1	Tone 1-related message
898	VoiceNotify_DCF_Tone2	Tone 2-related message
899	VoiceNotify_DCF_ETC	Other messages

900

901 1.4.2 CCP Notification type

902	CELF_CS_CCP_CALLING_START_REQ	Notification of starting display during CCP outgoing
903	CELF_CS_CCP_CALLED_START_IND	Notification of starting display during CCP incoming
904	CELF_CS_CCP_CALLING_ALERTING_IND	Notification of starting display during CCP calling
905	CELF_CS_CCP_CONNECT_START_RSP	Notification of starting display during CCP connection
906		
907	CELF_CS_CCP_CONNECT_START_IND	Notification of starting display during CCP communication
908		
909	CELF_CS_CCP_RELEASE_IND	Notification of ending CCP display
910	CELF_CS_CCP_DISCONNECT_REQ	Notification of starting CCP disconnection (on a mobile device) display
911		
912	CELF_CS_CCP_DISCONNECT_START_IND	Notification of starting CCP disconnection (on a network) display
913		
914	CELF_CS_CCP_CALLING_REJ_IND	Notification of rejecting CCP outgoing
915	CELF_CS_CCP_HOLD_CNF	Notification of CCP hold
916	CELF_CS_CCP_RETRIEVE_CNF	Notification of releasing CCP hold
917	CELF_CS_CCP_CALLING_SETUP_REQ	Notification of registering CCP outgoing call history
918	CELF_CS_CCP_CALLED_REJ_REQ	Notification of registering CCP absence incoming call history
919		
920	CELF_CS_CCP_CALLED_SETUP_RSP	Notification of registering CCP incoming call history
921	CELF_CS_CCP_RGT_START	Notification of CCP RGT start
922	CELF_CS_CCP_RGT_STOP	Notification of CCP RGT stop
923	CELF_CS_CCP_HRGT_START	Start notification of incoming of a CCP hold call
924	CELF_CS_CCP_HRGT_STOP	Stop notification of incoming of a CCP hold call
925	CELF_CS_CCP_DST_START	Notification of CCP DST start
926	CELF_CS_CCP_DST_STOP	Notification of CCP DST stop
927	CELF_CS_CCP_RBT_START	Notification of CCP RBT start
928	CELF_CS_CCP_RBT_STOP	Notification of CCP RBT stop
929	CELF_CS_CCP_BT_START	Notification of CCP BT start

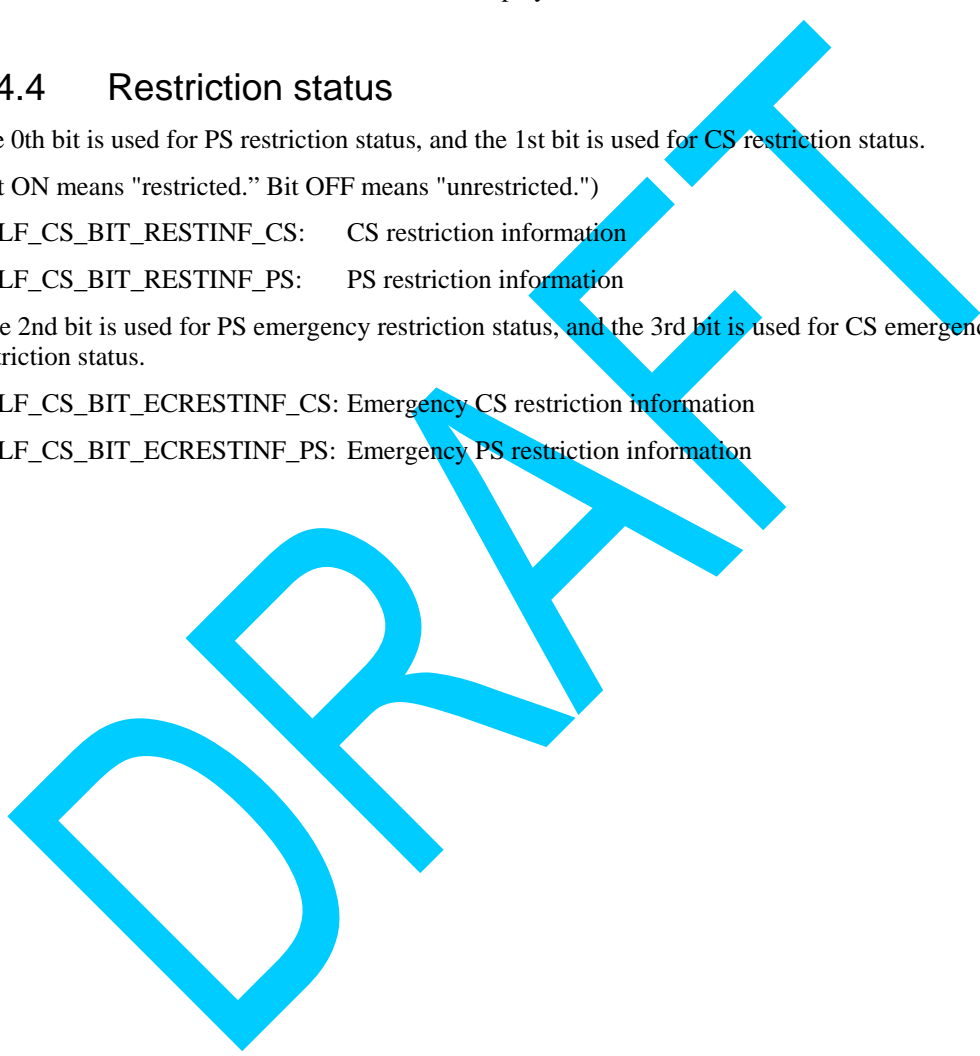
930 CELF_CS_CCP_CWT_START Notification of CCP CWT start
931 CELF_CS_CCP_CWT_STOP Notification of CCP CWT stop
932 CELF_CS_CCP_REJECT_ASK Inquiry report of rejecting a CCP CS incoming call
933

1.4.3 Notification type

934
935 CELF_CS_RSMP_REST_STA: Restriction display start notification
936 CELF_CS_RSMP_REST_END: Restriction display end notification
937

1.4.4 Restriction status

938
939 The 0th bit is used for PS restriction status, and the 1st bit is used for CS restriction status.
940 (Bit ON means "restricted." Bit OFF means "unrestricted.")
941 CELF_CS_BIT_RESTINF_CS: CS restriction information
942 CELF_CS_BIT_RESTINF_PS: PS restriction information
943 The 2nd bit is used for PS emergency restriction status, and the 3rd bit is used for CS emergency
944 restriction status.
945 CELF_CS_BIT_ECRESTINF_CS: Emergency CS restriction information
946 CELF_CS_BIT_ECRESTINF_PS: Emergency PS restriction information
947
948
949



2. Start Notification

2.1 Symbol: `self_mp_cs_notification_start`

2.1.1 Syntax

```
CelfMpStatus self_mp_cs_notification_start (  
    CelfMpAppID      app_id,  
    CelfMpCsNotifySet event_set,  
    CelfMpCallback   callback_func);
```

2.1.2 Argument

Name: `app_id`

Type: `CelfMpAppID`

I/O: I

Description:

Application identifier.

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:

`SELF_MP_CS_CLASS_COM_STATUS`: Voice communication status notification

`SELF_MP_CS_CLASS_TLK_TIME`: Call duration notification

`SELF_MP_CS_CLASS_DISC_CAUSE`: Disconnection cause notification

`SELF_MP_CS_CLASS_FW_RESULT`: Call forwarding result notification

`SELF_MP_CS_CLASS_OFFHK_TO`: Off-hook originating timeout notification

A callback **may** be registered for all classes of events using special event class

`SELF_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

986 Description:

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

988

989 2.1.3 Return Value

990 Type: `CelfMpStatus`

991 I/O: `O`

992 Description:

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

994 `CELF_MP_STATUS_OK`: successful completion

995 `CELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

996 `CELF_MP_STATUS_EVENT_SET_ERR`: Notification event set is not valid

997 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

998

999 2.1.4 Include File

1000 `/usr/include/celf/mp_cs.h`

1001

1002 2.1.5 Functional Description

1003 This function is used to start notification callbacks for events related to circuit switched communication.

1004 Events from a registered class **shall** cause the registered callback function to be called when the event
1005 occurs for the application identified by `app_id`. If a class of events does not have a registered callback
1006 function, no callback **shall** occur for those events.

1007

1008 The event structure in section 0.1.1 **must** be used and the value subtype **shall be set to**
1009 **“VoiceNotify_ConnInfo”**.

1010

1011

1012 3. Stop Notification

1013 3.1 Symbol: `celf_mp_cs_notification_stop`

1014 3.1.1 Syntax

```
1015 CelfMpStatus celf_mp_cs_notification_stop (  
1016     CelfMpAppID  app_id,  
1017     CelfMpNotifySet event_set);
```

1018 3.1.2 Argument

1019 Name: `app_id`

1020 Type: `CelfMpAppID`

1021 I/O: `I`

1022 Description:

1023 Application identifier.

1024

1025 Name: `event_set`

1026 Type: `CelfMpCsNotifySet`

1027 I/O: `I`

1028 Description:

1029 Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class
1030 **may** be registered to have a callback function called when the event occurs for the application identified by
1031 `app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

1032

1033 The event classes are defined as follows:

1034 `CELF_MP_CS_CLASS_COM_STATUS`: Voice communication status notification

1035 `CELF_MP_CS_CLASS_TLK_TIME` : Call duration notification

1036 `CELF_MP_CS_CLASS_DISC_CAUSE`: Disconnection cause notification

1037 `CELF_MP_CS_CLASS_FW_RESULT` : Call forwarding result notification

1038 `CELF_MP_CS_CLASS_OFFHK_TO` : Off-hook originating timeout notification

1039

1040 A callback **may** be registered for all classes of events using special event class

1041 `CELF_MP_CS_CLASS_ALL`, however to reduce overhead it is recommended that only the needed event
1042 classes **should** be registered.

1043

1044 3.1.3 Return Value

1045 Type: `CelfMpStatus`

1046 I/O: `O`

1047 Description:

1048 `celf_mp_cs_notification_stop()` **shall** return one of the following values:

- 1049 CELF_MP_STATUS_OK: successful completion
- 1050 CELF_MP_STATUS_APP_ID_ERR: Application ID is not valid.
- 1051 CELF_MP_STATUS_EVENT_SET_ERR: Notification event set is not valid
- 1052 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1053

1054 3.1.4 Include File

1055 `/usr/include/celf/mp_cs.h`

1056

1057 3.1.5 Functional Description

1058 This function stops voice communication related event reporting.

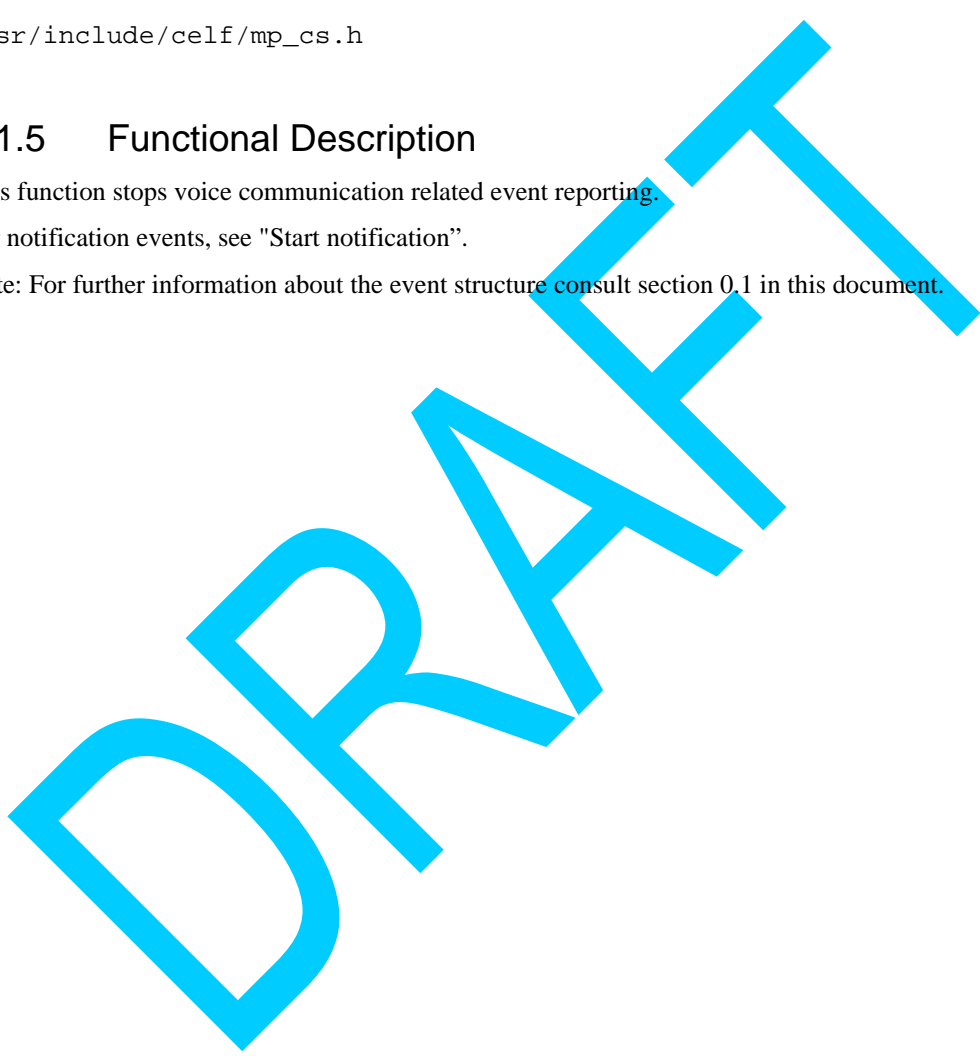
1059 For notification events, see "Start notification".

1060 Note: For further information about the event structure consult section 0.1 in this document.

1061

1062

1063



4. Get Voice Communication Status

4.1 Symbol: `celf_mp_cs_get_com_status`

4.1.1 Syntax

```
CelfMpStatus celf_mp_cs_get_com_status (  
    CelfMpAppID app_id);
```

4.1.2 Argument

Name: `app_id`

Type: `CelfMpAppID`

I/O: `I`

Description:

Application identifier.

4.1.3 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.

4.1.4 Include File

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

4.1.5 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.

1091 5. Get Connection Information to Other Party

1092 5.1 Symbol: `celf_mp_cs_get_con_info_ref`

1093 5.1.1 Syntax

```
1094 CelfMpStatus celf_mp_cs_get_con_info_ref (  
1095     CelfMpAppID      app_id,  
1096     CelfMpCallNo     call_no,  
1097     CelfMpConnectInfo connect_inf_p);
```

1098 5.1.2 Argument

1099 Name: `app_id`

1100 Type: `CelfMpAppID`

1101 I/O: `I`

1102 Description:

1103 Application identifier.

1104

1105 Name: `call_no`

1106 Type: `CelfMpCallNo`

1107 I/O: `I`

1108 Description:

1109 Call reference (0 to 255).

1110

1111 Name: `connect_inf_p`

1112 Type: `CelfMpConnectInfo`

1113 I/O: `O`

1114 Description:

1115 Pointer to the connection destination information. See section 0.1 for details.

1116

1117 5.1.3 Return Value

1118 Type: `CelfMpStatus`

1119 I/O: `O`

1120 Description:

1121 `celf_mp_cs_get_con_info_ref()` **shall** return one of the values defined:

1122 `CELF_MP_STATUS_OK:` successful completion

1123 `CELF_MP_STATUS_APP_ID_ERR:` Application ID is not valid.

1124 `CELF_MP_STATUS_CALL_NO_ERR:` Call number is not valid

1125 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

1126

1127 5.1.4 Include File

1128 /usr/include/celf/mp_cs.h

1129

1130 5.1.5 Functional Description

1131 This function refers to the connection information to other party specified call reference

1132 Without the monitoring the voice communication, it is possible to get the connection information

1133

1134 In the following cases, The result (STS) is set CELF_CS_ERR.

1135 1. The call specified by call reference does not exist.

1136 2. Others parameter Error.

DRAFT

1137 **6. Get Call Duration**

1138 **6.1 Symbol: celf_mp_cs_get_call_duration**

1139 **6.1.1 Syntax**

1140 CelfMpTime celf_mp_cs_get_call_duration (
1141 CelfMpAppID app_id);

1142 **6.1.2 Argument**

1143 Name: app_id

1144 Type: CelfMpAppID

1145 I/O: I

1146 Description:

1147 Application identifier.

1148

1149 **6.1.3 Return Value**

1150 Type: CelfMpTime

1151 I/O: O

1152 Description:

1153 celf_mp_cs_get_call_duration() shall return the current call duration in seconds.

1154

1155 **6.1.4 Include File**

1156 /usr/include/celf/mp_cs.h

1157

1158 **6.1.5 Functional Description**

1159 This function gets the call duration on the current call.

1160 The call duration is counted by the voice communication service.

1161 When no call exists, the function returns zero.

1162

1163 7. Off-Hook Notification

1164 7.1 Symbol: `celf_mp_cs_notification_off_hook`

1165 7.1.1 Syntax

```
1166 CelfMpStatus celf_mp_cs_notification_off_hook (  
1167     CelfMpAppID  app_id,  
1168     CelfMpCsBtype com_type,  
1169     CelfMpCsOffHk option);
```

1170 7.1.2 Argument

1171 Name: `app_id`

1172 Type: `CelfMpAppID`

1173 I/O: `I`

1174 Description:

1175 Application identifier.

1176

1177 Name: `com_type`

1178 Type: `celfCsBtype`

1179 I/O: `I`

1180 Description:

1181 Communication type as defined in section 0.1.

1182

1183 Name: `option`

1184 Type: `CelfMpCsOffHk`

1185 I/O: `I`

1186 Description:

1187 One the following options **shall** be set:

1188 `CEL_F_CS_OFFHK_AUTO` Automatic transmission

1189 `CEL_F_CS_OFFHK_MANUAL` Manual transmission

1190

1191 7.1.3 Return Value

1192 Type: `CelfMpStatus`

1193 I/O: `O`

1194 Description:

1195 `celf_mp_cs_notification_off_hook()` **shall** return one of the values defined:

1196 `CEL_F_MP_STATUS_OK`: successful completion

1197 `CEL_F_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1198 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1199 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1200

1201 7.1.4 Include File

1202 /usr/include/celf/mp_cs.h

1203

1204 7.1.5 Functional Description

1205 This function receives the request of off-hook.

1206

1207 The term “off-hook” refers to the user first presses the "dial" button, then enters the number to dial.

1208

1209 By this function,

1210 (1) When the mobile phone is in the wait (standby) status, the dial tone (DT) sounds and it is possible to
1211 input dial number, or

1212 (2) When the input of dial number is completed, the mobile phone starts the originating.

1213 Because the function is an immediate return function, to confirm the complete result, including the
1214 negotiation with the network, `celf_mp_cs_notification_status()` shall be used to obtain the
1215 communication status.

1216

1217 The process at timer timeout (five seconds) varies depending on the specification of “option”.

1218 This timer count starts at the last dial inputting.

1219 (1) When the "option" is `CELF_CS_OFFHK_AUTO` (automatic originating)

1220 Automatic originating operation is immediately performed by the dials, which were already input in
1221 "Dial ".

1222 (2) When the "option" is `CELF_CS_OFFHK_MANUAL` (manual originating)

1223 It is notified timeout to an application, and waits for the notification of originating from
1224 the application. ("Complete dial" or "On-hook originating")

1225 Timeout is notified by monitoring "Off-hook originating timeout notification" in "Start
1226 voice communication status monitoring".

1227

1228 When a mobile phone is moved to low voltage mode, a low voltage notification is sent.

1229 During low voltage, when the communication status is other than the under standby, this Off-hook is
1230 disabled.

1231

1232 If an incoming call arrives during off-hook, this Off-hook is cancelled.

1233

1234 In case of using the subaddress, it should be use the function "On-hook originating".

1235 **8. Disconnect**

1236 **8.1 Symbol: celf_mp_cs_disconnect**

1237 **8.1.1 Syntax**

1238 CelfMpStatus celf_mp_cs_disconnect (
1239 CelfMpAppID app_id
1240 CelfMpCsBtype com_type);

1241 **8.1.2 Argument**

1242 Name: app_id

1243 Type: CelfMpAppID

1244 I/O: I

1245 Description:

1246 Application identifier.

1247

1248 Name: com_type

1249 Type: CelfMpCsBtype

1250 I/O: I

1251 Description:

1252 Communication type as defined in section 0.1.

1253

1254 **8.1.3 Return Value**

1255 Type: CelfMpStatus

1256 I/O: O

1257 Description:

1258 `celf_mp_cs_disconnect()` shall return one of the values defined:

1259 CELF_MP_STATUS_OK: successful completion

1260 CELF_MP_STATUS_APP_ID_ERR: Application ID is not valid.

1261 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1262 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1263

1264 **8.1.4 Include File**

1265 `/usr/include/celf/mp_cs.h`

1266

1267 **8.1.5 Functional Description**

1268 This function receives the request to disconnect the call.

1269

1270 Because the function is an immediate return function, to confirm the complete result, including the
1271 negotiation with the network, it should be issued "celf_mp_cs_notification_status()" to obtain the
1272 communication status.

1273

1274 An incoming call cannot be disconnected by this function. (Use "Reject incoming call")

1275

1276 If multiple calls exist, all calls are disconnected.

DRAFT

1277 **9. Dial**

1278 **9.1 Symbol: celf_mp_cs_dial**

1279 **9.1.1 Syntax**

```
1280 CelfMpStatus celf_mp_cs_dial (  
1281     CelfMpAppID  app_id  
1282     CelfMpCsBtype com_type,  
1283     CelfMpCsDialBuffer  dial_buf,  
1284     CelfMpCsDialLen    dial_len);
```

1285 **9.1.2 Argument**

1286 Name: app_id

1287 Type: CelfMpAppID

1288 I/O: I

1289 Description:

1290 Application identifier.

1291

1292 Name: com_type

1293 Type: CelfMpCsBtype

1294 I/O: I

1295 Description:

1296 Communication type as defined in section 0.1.

1297

1298 Name: dial_buf

1299 Type: CelfMpCsDialBuffer

1300 I/O: I

1301 Description:

1302 Dial data buffer address

1303

1304 Name: dial_len

1305 Type: CelfMpCsDialLen

1306 I/O: I

1307 Description:

1308 Dial data length

1309

1310 **9.1.3 Return Value**

1311 Type: CelfMpStatus

1312 I/O: O
1313 Description:
1314 `celf_mp_cs_dial()` shall return one of the values defined:
1315 CELF_MP_STATUS_OK: successful completion
1316 CELF_MP_STATUS_APP_ID_ERR: Application ID is not valid.
1317 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid
1318 CELF_MP_STATUS_ERR: Other unsuccessful completion.
1319

1320 9.1.4 Include File

1321 `/usr/include/celf/mp_cs.h`
1322

1323 9.1.5 Functional Description

1324 This function receives the sequence of dial number.
1325

1326 Because the function is an immediate return function, to confirm the complete result, including the
1327 negotiation with the network, it should be issued "`celf_mp_cs_notification_status()`" to obtain the
1328 communication status.
1329

1330 The dial data stores the following ASCII codes.

1331 1 : 0 x 31 2 : 0 x 32 3 : 0 x 33
1332 4 : 0 x 34 5 : 0 x 35 6 : 0 x 36
1333 7 : 0 x 37 8 : 0 x 38 9 : 0 x 39
1334 * : 0 x 2a 0 : 0 x 30 # : 0 x 23
1335

1336 When "Off-hook" is called, the mobile phone is in off-hook status.

1337 Under this off-hook status, the mobile phone starts an outgoing call with "Dial" and "Complete dial".

1338 Five seconds later from the last digit has been entered, the outgoing process starts automatically, when
1339 automatic transmission is specified in "Off-hook".

1340 When "Off-hook" is called, the mobile phone is in off-hook status.
1341

1342 Under this on-hook status, DTMF is sent, if the status is (a) the conversation or (b) the conversation and
1343 hold.

1344

10.Dial Complete

1345

10.1 Symbol: `celf_mp_cs_dial_end`

1346

10.1.1 Syntax

1347

```
CelfMpStatus celf_mp_cs_dial_end (
```

1348

```
    CelfMpAppID  app_id
```

1349

```
    CelfMpCsBtype com_type);
```

1350

10.1.2 Argument

1351

Name: `app_id`

1352

Type: `CelfMpAppID`

1353

I/O: `I`

1354

Description:

1355

Application identifier.

1356

1357

Name: `com_type`

1358

Type: `CelfMpCsBtype`

1359

I/O: `I`

1360

Description:

1361

Communication type as defined in section 0.1.

1362

1363

10.1.3 Return Value

1364

Type: `CelfMpStatus`

1365

I/O: `O`

1366

Description:

1367

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

1368

`CELF_MP_STATUS_OK:` successful completion

1369

`CELF_MP_STATUS_APP_ID_ERR:` Application ID is not valid.

1370

`CELF_MP_STATUS_COM_TYPE_ERR:` Communication type is not valid

1371

`CELF_MP_STATUS_ERR:` Other unsuccessful completion.

1372

1373

10.1.4 Include File

1374

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

1375

1376

10.1.5 Functional Description

1377

This function receives the request to end the dial entry.

1378

1379 Because this is an asynchronous function the service will return the result through a notification.

1380 `celf_mp_cs_notification_status()` shall be used to obtain the communication status.

1381 Under off-hook status, the mobile phone starts outgoing operation by calling this function with dial

1382 number, which was given by preceding function calls "Dial".

1383

1384 Under on-hook status, the calling this function is disabled.

1385

DRAFT

1386 **11.Response to Incoming Call**

1387 **11.1 Symbol: celf_mp_cs_call_rcv**

1388 **11.1.1 Syntax**

1389 CelfMpStatus celf_mp_cs_call_rcv (
1390 CelfMpAppID app_id
1391 CelfMpCsBtype com_type);

1392 **11.1.2 Argument**

1393 Name: app_id

1394 Type: CelfMpAppID

1395 I/O: I

1396 Description:

1397 Application identifier.

1398

1399 Name: com_type

1400 Type: CelfMpCsBtype

1401 I/O: I

1402 Description:

1403 Communication type as defined in section 0.1.

1404

1405 **11.1.3 Return Value**

1406 Type: CelfMpStatus

1407 I/O: O

1408 Description:

1409 `celf_mp_cs_call_rcv()` shall return one of the values defined:

1410 CELF_MP_STATUS_OK: successful completion

1411 CELF_MP_STATUS_APP_ID_ERR: Application ID is not valid.

1412 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1413 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1414

1415 **11.1.4 Include File**

1416 `/usr/include/celf/mp_cs.h`

1417

1418 **11.1.5 Functional Description**

1419 This function receives the request to process an incoming call.

1420

1421 Because the function is an immediate return function, to confirm the complete result, including the
1422 negotiation with the network, it should be issued "celf_mp_cs_notification_status()" to obtain the
1423 communication status.

1424

1425 One of the following operations is performed depending on the mobile phone status.

1426 Under incoming : Responds to the incoming call.

1427 Under response hold : Responds to the response hold call

1428 Others : Disabled

1429

1430 If the mobile phone is in low voltage mode, this function is disabled.

1431

1432 To respond to the incoming call in the status, "under conversation and incomings", use "Reject incoming
1433 call".

DRAFT

1434 **12.Forward Incoming Call**

1435 **12.1 Symbol: celf_mp_cs_call_forward**

1436 **12.1.1 Syntax**

1437 CelfMpStatus celf_mp_cs_call_forward (
1438 CelfMpCsBtype com_type);

1439 **12.1.2 Argument**

1440

1441 Name: com_type

1442 Type: CelfMpCsBtype

1443 I/O: I

1444 Description:

1445 Communication type as defined in section 0.1.

1446

1447 **12.1.3 Return Value**

1448 Type: CelfMpStatus

1449 I/O: O

1450 Description:

1451 celf_mp_cs_call_forward() shall return one of the values defined:

1452 CELF_MP_STATUS_OK: successful completion

1453 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1454 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1455

1456 **12.1.4 Include File**

1457 /usr/include/celf/mp_cs.h

1458

1459 **12.1.5 Functional Description**

1460 This function receives the request to forward an incoming call.

1461

1462 Because the function is an immediate return function, to confirm the complete result, including the
1463 negotiation with the network, it should be issued "celf_mp_cs_notification_status()" to obtain the
1464 communication status.

1465

1466 The incoming call is forwarded when the communication status is (a)under the incoming, (b)under
1467 conversation and incoming, or (c)under hold and incoming.

1468

1469 If the forwarding fails, incoming call is continued between other party and this phone.
1470

DRAFT

1471

13. Forward to Voice Mail System

1472

13.1 Symbol: `celf_mp_cs_call_forward_voice_msg`

1473

13.1.1 Syntax

1474

`CelfMpStatus celf_mp_cs_call_forward_voice_msg (`

1475

`CelfMpCsBtype com_type);`

1476

13.1.2 Argument

1477

1478

Name: `com_type`

1479

Type: `CelfMpCsBtype`

1480

I/O: `I`

1481

Description:

1482

Communication type as defined in section 0.1.

1483

1484

13.1.3 Return Value

1485

Type: `CelfMpStatus`

1486

I/O: `O`

1487

Description:

1488

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

1489

`CELf_MP_STATUS_OK`: successful completion

1490

`CELf_MP_STATUS_COM_TYPE_ERR`: Communication type is not valid

1491

`CELf_MP_STATUS_ERR`: Other unsuccessful completion.

1492

1493

13.1.4 Include File

1494

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

1495

1496

13.1.5 Functional Description

1497

This function receives the request to forward a call to a voice mail system.

1498

1499

Because the function is an immediate return function, to confirm the complete result, including the negotiation with the network, it should be issued "`celf_mp_cs_notification_status()`" to obtain the communication status.

1500

1501

1502

1503

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.

1504

1505

1506 If the forwarding fails, incoming call is continued between other party and this phone.

DRAFT

1507

14.Call Hold

1508

14.1 Symbol: `celf_mp_cs_call_hold`

1509

14.1.1 Syntax

1510

`CelfMpStatus celf_mp_cs_call_hold (`

1511

`CelfMpCsBtype com_type);`

1512

14.1.2 Argument

1513

1514 Name: `com_type`

1515 Type: `CelfMpCsBtype`

1516 I/O: `I`

1517 Description:

1518 Communication type as defined in section 0.1.

1519

14.1.3 Return Value

1521 Type: `CelfMpStatus`

1522 I/O: `O`

1523 Description:

1524 `celf_mp_cs_call_hold()` shall return one of the values defined:

1525 `CELf_MP_STATUS_OK`: successful completion

1526 `CELf_MP_STATUS_COM_TYPE_ERR`: Communication type is not valid

1527 `CELf_MP_STATUS_ERR`: Other unsuccessful completion.

1528

14.1.4 Include File

1530 `/usr/include/celf/mp_cs.h`

1531

14.1.5 Functional Description

1533 This function receives the requests response hold.

1534

1535 Because the function is an immediate return function, to confirm the complete result, including the
1536 negotiation with the network, it should be issued "`celf_mp_cs_notification_status()`" to obtain the
1537 communication status.

1538

1539 This response hold is performed for an incoming call, only when the communication status is under
1540 incoming.

1541

1542 To release response hold (move to the under conversation status) call "Response to an incoming call".

1543

1544

DRAFT

1545 **15.Call Reject**

1546 **15.1 Symbol: celf_mp_cs_call_reject**

1547 **15.1.1 Syntax**

1548 CelfMpStatus celf_mp_cs_call_reject (
1549 CelfMpCsBtype com_type);

1550 **15.1.2 Argument**

1551

1552 Name: com_type

1553 Type: CelfMpCsBtype

1554 I/O: I

1555 Description:

1556 Communication type as defined in section 0.1.

1557

1558 **15.1.3 Return Value**

1559 Type: CelfMpStatus

1560 I/O: O

1561 Description:

1562 celf_mp_cs_call_reject() shall return one of the values defined:

1563 CELF_MP_STATUS_OK: successful completion

1564 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1565 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1566

1567 **15.1.4 Include File**

1568 /usr/include/celf/mp_cs.h

1569

1570 **15.1.5 Functional Description**

1571 This function receives the request to reject an incoming call.

1572

1573 Because the function is an immediate return function, to confirm the complete result, including the
1574 negotiation with the network, it should be issued "celf_mp_cs_notification_start()" to obtain the
1575 communication status.

1576

1577 The operation for each communication status is as follows:

1578 Under incoming: Rejects an incoming call

1579 Under conversation and incoming: Rejects an incoming call

- 1580 Under hold and incoming: Rejects an incoming call
- 1581 Under conversation, hold, and incoming: Rejects an incoming call
- 1582
- 1583
- 1584

DRAFT

1585 **16.Multi Party Call**

1586 **16.1 Symbol: celf_mp_cs_mp_call**

1587 **16.1.1 Syntax**

```
1588 CelfMpStatus celf_mp_cs_mp_call (  
1589     CelfMpCsBtype com_type,  
1590     CelfMpCsMop mode,  
1591     CelfMpCsCallRef call_reference);
```

1592 **16.1.2 Argument**

1593

1594 Name: com_type

1595 Type: CelfMpCsBtype

1596 I/O: I

1597 Description:

1598 Communication type as defined in section 0.1.

1599

1600 Name: mode

1601 Type: CelfMpCsMop

1602 I/O: I

1603 Description:

1604 Operation type

1605 CELF_CS_MOP_RSV_DISC: Disconnect the hold call

1606 CELF_CS_MOP_DISC_AND_RSP: Response after disconnection

1607 CELF_CS_MOP_RSV_AND_RSP: Response after hold (including operation for switching a call)

1608 CELF_CS_MOP_CR_DISC: Disconnect call specified by the call reference

1609

1610 Name: call_reference

1611 Type: CelfMpCsCallRef

1612 I/O: I

1613 Description:

1614 Call reference of the call to be disconnected

1615 Valid only if CELF_CS_MOP_CR_DISC is specified for the second argument.

1616

1617

1618 **16.1.3 Return Value**

1619 Type: CelfMpStatus

1620 I/O: O
1621 Description:
1622 `celf_mp_cs_mp_call()` **shall** return one of the values defined:
1623 CELF_MP_STATUS_OK: successful completion
1624 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid
1625 CELF_MP_STATUS_ERR: Other unsuccessful completion.
1626

1627 16.1.4 Include File

1628 `/usr/include/celf/mp_cs.h`

1629

1630 16.1.5 Functional Description

1631 This function receives the request to operate for each call, when communication is made with multiple
1632 calls.

1633

1634 The operation is as follows depending on CELF_CS_MOP:

1635 - CELF_CS_MOP_RSV_DISC

1636 If a hold call exists, this hold call is disconnected.

1637

1638 - CELF_CS_MOP_DISC_AND_RSP

1639 If a conversation call exists and if another call status is incoming or hold, the conversation call transits to
1640 disconnect status and another call transits to conversation status.

1641 See detail below.

1642 (1) Under conversation and incoming

1643 This status is that 1st call is in conversation, and 2nd call is incoming.

1644 The result is that 1st call is released, and 2nd call is conversation.

1645 (2) Under conversation and hold

1646 This status is that 1st call is in conversation, and 2nd call is hold.

1647 The result is that 1st call is released, and 2nd call is conversation.

1648 (3) Under conversation, hold, and incoming

1649 This status is that 1st call is in conversation, that 2nd call is hold, and that 3rd call is incoming.

1650 The result is that 1st call is released, that 2nd call maintains hold, and that 3rd call is conversation.

1651 (4) Under response hold

1652 This status is not changed.

1653

1654 - CELF_CS_MOP_RSV_AND_RSP

1655 If a conversation call exists and if another call status is incoming or hold,

1656 the conversation call transits to hold status and another call transits to conversation status.

Classification: *Circuit Switched Service*

1657 See detail below.

1658 (1) Under conversation and incoming

1659 This status is that 1st call is in conversation, and 2nd call is incoming.

1660 The result is that 1st call is hold, and 2nd call is conversation.

1661 (2) Under conversation and hold

1662 This status is that 1st call is in conversation, and 2nd call is hold.

1663 The result is that 1st call is hold, and 2nd call is conversation.

1664 (3) Under conversation, hold, and incoming

1665 This status is that 1st call is in conversation, that 2nd call is hold, and that 3rd call is incoming.

1666 The result is that 1st call is hold, that 2nd call is in conversation, that 3rd call maintains incoming.

1667 (4) Under response hold

1668 This status is that 1st call is hold.

1669 The result is that 1st call is in conversation.

1670

1671 - CELF_CS_MOP_CR_DISC It is disconnect the call specified the call reference.

1672

1673 Because the function is an immediate return function, to confirm the complete result, including the

1674 negotiation with the network, it should be issued "celf_mp_cs_notification_start()" to obtain the

1675 communication status.

1676

17. On-Hook Originating

1677

17.1 Symbol: `celf_mp_cs_originating_on_hook`

1678

17.1.1 Syntax

1679

`CelfMpStatus celf_mp_cs_originating_on_hook (`

1680

`CelfMpAppID app_id,`

1681

`CelfMpCsConReq con_req);`

1682

17.1.2 Argument

1683

1684 Name: `app_id`

1685 Type: `CelfMpAppID`

1686 I/O: `I`

1687 Description:

1688 Application identifier.

1689

1690 Name: `con_req`

1691 Type: `CelfMpCsConReq`

1692 I/O: `I`

1693 Description:

1694 Communication request type as defined in section 0.1.

1695

17.1.3 Return Value

1697 Type: `CelfMpStatus`

1698 I/O: `O`

1699 Description:

1700 `celf_mp_cs_call_reject()` shall return one of the values defined:

1701 `CELf_CS_OK`: Normal

1702 `CELf_CS_ONHOOK_DENY`: On-hook originating is impossible.

1703 `CELf_CS_ONHOOK_STATUS_ERR`: Error due to communication conflict

1704 `CELf_CS_ONHOOK_OB_CR`: Excess of the maximum number of calls

1705 `CELf_CS_ERR`: Abnormal

1706

17.1.4 Include File

1708 `/usr/include/celf/mp_cs.h`

1709

1710 **17.1.5 Functional Description**

1711 This function receives the request to start an outgoing call with the specified dial number.

1712 The communication status should be Standby.

1713

1714 The dial number is specified by "dial_buf" and "subaddr_buf" in the "con_req" structure.

1715

1716 If the character string, "184" or "186", is placed at the head of dial data, this character string is deleted.

1717 Whether the originating dial number is notified or not, it is identified by "notice".

1718

1719 The dial data and subaddress stores the following ASCII codes.

1720 1 : 0 x 31 2 : 0 x 32 3 : 0 x 33

1721 4 : 0 x 34 5 : 0 x 35 6 : 0 x 36

1722 7 : 0 x 37 8 : 0 x 38 9 : 0 x 39

1723 * : 0 x 2a 0 : 0 x 30 # : 0 x 23

1724

1725 Because the function is an immediate return function, to confirm the complete result, including the
1726 negotiation with the network, it should be issued "celf_mp_cs_notification_start()" to obtain the
1727 communication status.

1728

1729 The originating request during low voltage is disabled.

1730 **18.Get Call Reference**

1731 **18.1 Symbol: celf_mp_cs_get_call_reference**

1732 **18.1.1 Syntax**

1733 CelfMpStatus celf_mp_cs_get_call_reference (
1734 CelfMpAppID app_id
1735 CelfMpCsChanNum channel_num);

1737 **18.1.2 Argument**

1738 Name: app_id

1739 Type: CelfMpAppID

1740 I/O: I

1741 Description:

1742 Application identifier.

1743

1744 Name: channel_num

1745 Type: CelfMpCsChanNum

1746 I/O: O

1747 Description:

1748 Channel number information as defined in section 0.1.

1749

1750 **18.1.3 Return Value**

1751 Type: CelfMpStatus

1752 I/O: O

1753 Description:

1754 celf_mp_cs_get_call_reference() **shall** return one of the values defined:

1755 CELF_MP_STATUS_OK: successful completion

1756 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1757

1758 **18.1.4 Include File**

1759 /usr/include/celf/mp_cs.h

1760

1761 **18.1.5 Functional Description**

1762 This function gets the call reference in use.

1763

Classification: *Circuit Switched Service*

1764 A value within 0 to 255 is set to "ChanNum_00", "ChanNum_01" and "ChanNum_02". If channel is not
1765 used, CELF_CS_CHAN_NOUSE is set as the call reference.
1766
1767 Three channel corresponds to three call in multiple call.
1768

DRAFT

1769 **19.Start DCF message notification**

1770 **19.1 Symbol: celf_mp_cs_DCF_notification_start**

1771 **19.1.1 Syntax**

1772 CelfMpStatus celf_mp_cs_DCF_notification_start (

1773 CelfMpAppID app_id,

1774 CelfMpDCFSet event_set);

1775 **19.1.2 Argument**

1776 Name: app_id

1777 Type: CelfMpAppID

1778 I/O: I

1779 Description:

1780 Application identifier.

1781

1782 Name: event_set

1783 Type: CelfMpCsDCFSet

1784 I/O: I

1785 Description:

1786 Notification event set. Events that are classified as belonging to one of the CelfMpCsDCFSet class **may**
1787 be registered to have a callback function called when the event occurs for the application identified by
1788 app_id. Classes of events are enabled by setting the corresponding bit in event_set:

1789

1790 The event classes are defined as follows:

1791 CELF_MP_CS_DCF_DISP Display-related message

1792 CELF_MP_CS_DCF_HISTORY History-related message

1793 CELF_MP_CS_DCF_TONE1 Tone 1-related message

1794 CELF_MP_CS_DCF_TONE2 Tone 2-related message

1795 CELF_MP_CS_DCF_ETC Other messages

1796 CELF_MP_CS_CLASS_ALL All notified

1797

1798 A callback **may** be registered for all classes of events using special event class

1799 CELF_MP_CS_CLASS_ALL, however to reduce overhead it is recommended that only the needed event
1800 classes **should** be registered.

1801

1802 Name: callback_func

1803 Type: CelfMpCallback

1804 I/O: I

1805 Description:

1806 The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

1807 19.1.3 Return Value

1808 Type: `CelfMpStatus`

1809 I/O: `O`

1810 Description:

1811 `celf_mp_cs_DCF_notification_start ()` **shall** return one of the values defined:

1812 `CELf_MP_STATUS_OK`: successful completion

1813 `CELf_MP_STATUS_ERR`: Other unsuccessful completion.

1814

1815 19.1.4 Include File

1816 `/usr/include/celf/mp_cs.h`

1817

1818 19.1.5 Functional Description

1819 This function starts the monitoring the DCF message on the voice communication or AV communication.

1820

1821 The occurrence of the event is notified to the application, specified by `app_id`.

1822

1823 The messages to be notified are described below.

1824

1825 Display-related message:

1826 -Notification of starting display during CCP outgoing

1827 -Notification of starting display during CCP incoming

1828 -Notification of starting display during CCP calling

1829 -Notification of starting display during CCP connecting

1830 -Notification of starting display during CCP communication

1831 -Notification of ending CCP That is to notifies of release of a CCP call.

1832 -Notification of starting CCP disconnection (on the mobilephone) display

1833 -Notification of starting display of CCP disconnection (on the network) display

1834 -Notification of rejecting CCP outgoing

1835 -Notification of CCP hold

1836 -Notification of releasing CCP hold

1837

1838 History-related message:

1839 -Notification of registering CCP outgoing call history

1840 -Notification of registering CCP absence incoming call history

CE Linux Forum Technical Document

Classification: *Circuit Switched Service*

- 1841 -Notification of registering CCP incoming call history
- 1842
- 1843 Tone 1-related message:(Tone sounding on the AP layer)
- 1844 -Notification of CCP RGT start
- 1845 -Notification of CCP RGT stop
- 1846 -Start report of incoming of a CCP hold call
- 1847 -Stop report of incoming of a CCP hold call
- 1848
- 1849 Tone 2-related message:(Tone sounding by the voice communication service)
- 1850 -Notification of CCP DST start
- 1851 -Notification of CCP DST stop
- 1852 -Notification of CCP RBT start
- 1853 -Notification of CCP RBT stop
- 1854 -Notification of CCP BT start
- 1855 -Notification of CCP CWT start
- 1856 -Notification of CCP CWT stop
- 1857
- 1858 Other messages:
- 1859 -Inquiry report of rejecting a CCP CS incoming call

DRAFT

1860 **20.Stop DCF message notification**

1861 **20.1 Symbol: celf_mp_cs_DCF_notification_stop**

1862 **20.1.1 Syntax**

```
1863 CelfMpStatus celf_mp_cs_DCF_notification_stop (  
1864     CelfMpAppID  app_id  
1865     CelfMpDCFSet event_set);
```

1866 **20.1.2 Argument**

1867 Name: app_id

1868 Type: CelfMpAppID

1869 I/O: I

1870 Description:

1871 Application identifier.

1872

1873 Name: event_set

1874 Type: CelfMpCsDCFSet

1875 I/O: I

1876 Description:

1877 Notification event set. Events that are classified as belonging to one of the CelfMpCsDCFSet class.
1878 Classes of events are enabled by setting the corresponding bit in event_set:

1879

1880 The event classes are defined as follows:

1881 CELF_MP_CS_DCF_DISP Display-related message

1882 CELF_MP_CS_DCF_HISTORY History-related message

1883 CELF_MP_CS_DCF_TONE1 Tone 1-related message

1884 CELF_MP_CS_DCF_TONE2 Tone 2-related message

1885 CELF_MP_CS_DCF_ETC Other messages

1886 CELF_MP_CS_CLASS_ALL All notified

1887

1888

1889 **20.1.3 Return Value**

1890 Type: CelfMpStatus

1891 I/O: O

1892 Description:

1893 celf_mp_cs_DCF_notification_stop() **shall** return one of the values defined:

1894 CELF_MP_STATUS_OK: successful completion

1895 CELF_MP_STATUS_ERR: Other unsuccessful completion.
1896

1897 20.1.4 Include File

1898 /usr/include/celf/mp_cs.h

1899

1900 20.1.5 Functional Description

1901 This function stops notifying of the DCF message on voice communication or AV communication.

DRAFT

1902 **21.Voice Message Notification**

1903 **21.1 Symbol: celf_mp_cs_voice_msg_notify**

1904 **21.1.1 Syntax**

1905 CelfMpStatus celf_mp_cs_voice_msg_notify (
1906 CelfMpCsRecMsg rec_status);

1907 **21.1.2 Argument**

1908
1909 Name: rec_status
1910 Type: CelfMpCsRecMsg
1911 I/O: I
1912 Description:
1913 CELF_CS_REC_MESSAGE_START: Start of a voice message
1914 CELF_CS_REC_MESSAGE_STOP: Stop of a voice message

1916 **21.1.3 Return Value**

1917 Type: CelfMpStatus
1918 I/O: O
1919 Description:
1920 celf_mp_cs_call_voice_msg_notify() **shall** return one of the values defined:
1921 CELF_MP_STATUS_OK: successful completion
1922 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid
1923 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1925 **21.1.4 Include File**

1926 /usr/include/celf/mp_cs.h

1928 **21.1.5 Functional Description**

1929 This function must be called before the communication state is changed to "under conversation."
1930 After the start notification, a stop notification **must** be issued, when the voice message is stopped.

1931 **22.Hold Tone Start**

1932 **22.1 Symbol: celf_mp_cs_hold_tone_start**

1933 **22.1.1 Syntax**

1934 CelfMpStatus celf_mp_cs_hold_tone_start (
1935 CelfMpAppID app_id);

1936 **22.1.2 Argument**

1937

1938 Name: app_id

1939 Type: CelfMpAppID

1940 I/O: I

1941 Description:

1942 Application identifier.

1943

1944 **22.1.3 Return Value**

1945 Type: CelfMpStatus

1946 I/O: O

1947 Description:

1948 celf_mp_cs_hold_tone_start() **shall** return one of the values defined:

1949 CELF_MP_STATUS_OK: successful completion

1950 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid

1951 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1952

1953 **22.1.4 Include File**

1954 /usr/include/celf/mp_cs.h

1955

1956 **22.1.5 Functional Description**

1957 This function starts to sound a hold tone during a call.

1958 **23.Hold Tone Stop**

1959 **23.1 Symbol: celf_mp_cs_hold_tone_stop**

1960 **23.1.1 Syntax**

1961 CelfMpStatus celf_mp_cs_hold_tone_stop (
1962 CelfMpAppID app_id);

1963 **23.1.2 Argument**

1964
1965 Name: app_id
1966 Type: CelfMpAppID
1967 I/O: I
1968 Description:
1969 Application identifier.

1971 **23.1.3 Return Value**

1972 Type: CelfMpStatus
1973 I/O: O
1974 Description:
1975 celf_mp_cs_hold_tone_stop() **shall** return one of the values defined:
1976 CELF_MP_STATUS_OK: successful completion
1977 CELF_MP_STATUS_COM_TYPE_ERR: Communication type is not valid
1978 CELF_MP_STATUS_ERR: Other unsuccessful completion.

1980 **23.1.4 Include File**

1981 /usr/include/celf/mp_cs.h

1983 **23.1.5 Functional Description**

1984 This function stops to sound a hold tone during a call.

1985

1986 **24. Get 64K / AV Communication Status**

1987 **24.1 Symbol: celf_mp_cs_get_UD_com_stat**

1988 **24.1.1 Syntax**

1989 CelfMpUDComStatus celf_mp_cs_get_UD_com_stat (
1990 void);

1991 **24.1.2 Argument**

1992
1993 None.

1995 **24.1.3 Return Value**

1996 Type: CelfMpUDComStatus

1997 I/O: O

1998 Description:

1999 celf_mp_cs_get_UD_com_stat() **shall** return one of the values defined:

2000 CELF_CS_UD_STOP: Under stop

2001 CELF_CS_UD_RUN: Under communication

2002 CELF_CS_UD_CALLED: Under incoming

2003 CELF_CS_UD_CALLING: Under outgoing

2004 CELF_CS_UD_DISCONNECT: Under disconnection

2005 CELF_CS_UD_CALLING_ALERT: Under calling

2006 CELF_CS_UD_HOLD: Under hold

2007

2008 **24.1.4 Include File**

2009 /usr/include/celf/mp_cs.h

2010

2011 **24.1.5 Functional Description**

2012 This function refers to the communication status of 64K communication or AV communication.

2013

2014

2015 **25. Get internal/external AV Communication Status**

2016 **25.1 Symbol: celf_mp_cs_get_AV_com_stat**

2017 **25.1.1 Syntax**

2018 CelfMpAVComStatus celf_mp_cs_get_AV_com_stat (
2019 void);

2020 **25.1.2 Argument**

2021 None.

2022

2023 **25.1.3 Return Value**

2024 Type: CelfMpAVComStatus

2025 I/O: O

2026 Description:

2027 `celf_mp_cs_get_AV_com_stat()` **shall** return one of the values defined:

2028 CELF_CS_AV_IN_STOP: Under stop

2029 CELF_CS_AV_IN_RUN: Under communication

2030 CELF_CS_AV_IN_CALLED: Under incoming

2031 CELF_CS_AV_IN_CALLING: Under outgoing

2032 CELF_CS_AV_IN_DISCONNECT: Under disconnection

2033 CELF_CS_AV_IN_CALLING_ALERT: Under calling

2034 CELF_CS_UD_IN_HOLD: Under hold

2035 CELF_CS_AV_OUT_STOP: Under stop

2036 CELF_CS_AV_OUT_RUN: Under communication

2037 CELF_CS_AV_OUT_CALLED: Under incoming

2038 CELF_CS_AV_OUT_CALLING: Under outgoing

2039 CELF_CS_AV_OUT_DISCONNECT: Under disconnection

2040 CELF_CS_AV_OUT_CALLING_ALERT: Under calling

2041 CELF_CS_UD_OUT_HOLD: Under hold

2042

2043 **25.1.4 Include File**

2044 `/usr/include/celf/mp_cs.h`

2045

2046 **25.1.5 Functional Description**

2047 This function refers to the communication status of internal or external AV communication.

2048

26. Get Communication Status

2049

26.1 Symbol: `celf_mp_cs_get_com_stat`

2050

26.1.1 Syntax

2051

`CelfMpStatus celf_mp_cs_get_com_stat (`

2052

`CelfMpAppID app_id,`

2053

`CelfMpCsRcvScene * rcv_scene);`

2054

26.1.2 Argument

2055

Name: `app_id`

2056

Type: `CelfMpAppID`

2057

I/O: `I`

2058

Description:

2059

Application identifier.

2060

Name: `rcv_scene`

2061

Type: `CelfMpCsRcvScene *`

2062

I/O: `O`

2063

Description:

2064

Incoming call type:

2065

`CELf_CS_RCV_SCENE_COMPETE_TRN` : Outgoing conflict

2066

`CELf_CS_RCV_SCENE_RSV_RETURN` : Incoming hold call

2067

`CELf_CS_RCV_SCENE_CALL_BACK` : Re-incoming

2068

`CELf_CS_RCV_SCENE_NORMAL`: Normal

2069

`CELf_CS_RCV_SCENE_NON`: Unset

2070

2071

26.1.3 Return Value

2072

Type: `CelfMpComStatus`

2073

I/O: `O`

2074

Description:

2075

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

2076

Current communication status

2077

`CELf_CS_COM_STATUS_WAIT`: Standby

2078

`CELf_CS_COM_STATUS_RCV`: Under incoming

2079

`CELf_CS_COM_STATUS_TRN`: Under outgoing

2080

`CELf_CS_COM_STATUS_DLV`: Under calling

2081

`CELf_CS_COM_STATUS_TLK`: Under conversation

2082

`CELf_CS_COM_STATUS_HLD`: Under response hold

2083 CELF_CS_COM_STATUS_DUMMY1: Under off-hook
2084 CELF_CS_COM_STATUS_RLS: Under release
2085 CELF_CS_COM_STATUS_TLK_RCV: Under conversation and incoming
2086 CELF_CS_COM_STATUS_TLK_TRN: Under conversation and outgoing
2087 CELF_CS_COM_STATUS_TLK_DLV: Under conversation and calling
2088 CELF_CS_COM_STATUS_TLK_RSV: Under conversation and hold
2089 CELF_CS_COM_STATUS_TLK_RLS: Under conversation and release
2090 CELF_CS_COM_STATUS_TLK_RSV_RCV: Under conversation, hold, and incoming
2091 CELF_CS_COM_STATUS_RCV_AV: Under incoming of an AV call
2092 CELF_CS_COM_STATUS_TRN_AV: Under outgoing of an AV call
2093 CELF_CS_COM_STATUS_DLV_AV: Under calling of an AV call
2094 CELF_CS_COM_STATUS_TLK_AV: Under conversation of an AV call
2095 CELF_CS_COM_STATUS_HLD_AV: Under response hold of an AV call
2096 CELF_CS_COM_STATUS_RLS_AV: Under release of an AV call
2097 CELF_CS_COM_STATUS_DUMMY2 : Under AV off-hook
2098 CELF_CS_ERR : Abnormal end

2099

2100 26.1.4 Include File

2101 /usr/include/celf/mp_cs.h

2102

2103 26.1.5 Functional Description

2104 This function returns the incoming call status, when the current call is (a) under incoming status or (b)
2105 under conversation and incoming status.

2106

2107 27.Start Line Status Notification

2108 27.1 Symbol: `celf_mp_cs_line_status_notification_start`

2109 27.1.1 Syntax

```
2110 CelfMpStatus celf_mp_cs_notification_start (  
2111     CelfMpAppID  app_id  
2112     CelfMpCsMtype event_set,  
2113     CelfMpCallback callback_func);
```

2114 27.1.2 Argument

2115 Name: `app_id`

2116 Type: `CelfMpAppID`

2117 I/O: `I`

2118 Description:

2119 Application identifier.

2120

2121 Name: `event_set`

2122 Type: `CelfMpCsMtype`

2123 I/O: `I`

2124 Description:

2125 Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class
2126 **may** be registered to have a callback function called when the event occurs for the application identified by
2127 `app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

2128 `CELF_CS_MONITOR_LINE_STATUS`: Line status change notification

2129 `CELF_CS_MONITOR_RESTRICT`: Restriction status change notification

2130 `CELF_CS_MONITOR_RSSI`: Receive level change notification

2131 `CELF_CS_MONITOR_ALL`: All notified

2132

2133 Name: `callback_func`

2134 Type: `CelfMpCallback`

2135 I/O: `I`

2136 Description:

2137 The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

2138

2139 27.1.3 Return Value

2140 Type: `CelfMpStatus`

2141 I/O: `O`

2142 Description:

2143 `celf_mp_cs_notification_start()` **shall** return one of the values defined:

- 2144 CELF_MP_STATUS_OK: successful completion
- 2145 CELF_MP_STATUS_APP_ID_ERR: Application ID is not valid.
- 2146 CELF_MP_STATUS_MON_TYPE_ERR: Monitor type is not valid
- 2147 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2148

2149 27.1.4 Include File

2150 `/usr/include/celf/mp_cs.h`

2151

2152 27.1.5 Functional Description

2153 This function starts the monitoring the line status.

2154 The occurrence of the event is notified to the application, specified by `app_id`.

2155 The events to be notified are described below.

2156

2157 1. Line status change notification:

2158 This event notifies that the line status is changed.

2159 The line status is the out-of-communication area status and the within-communication area.

2160

2161 2. Restriction status change notification:

2162 This event notifies that a restriction status is changed.

2163 The restriction means that the incoming call or the outgoing call is restricted by the network in case of
2164 traffic congestion.

2165

2166 3. Receive level change notification:

2167 This event notifies that the receive level is changed.

2168 The receive level is the intensity of electromagnetic wave. The intensity is four level, high, mid, low and
2169 zero (out of area).

2170

2171 See section 0.1 for structure definitions and values.

2172

2173 28. Stop Line Status Notification

2174 28.1 Symbol: `celf_mp_cs_line_status_notification_stop`

2175 28.1.1 Syntax

```
2176 CelfMpStatus celf_mp_cs_notification_stop (  
2177     CelfMpAppID  app_id  
2178     CelfMpCsMtype event_set);
```

2179 28.1.2 Argument

2180 Name: `app_id`

2181 Type: `CelfMpAppID`

2182 I/O: `I`

2183 Description:

2184 Application identifier.

2185

2186 Name: `event_set`

2187 Type: `CelfMpCsMtype`

2188 I/O: `I`

2189 Description:

2190 Mask of the events for which reporting is to be stopped.

2191 Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class
2192 **may** be registered to have a callback function called when the event occurs for the application identified by
2193 `app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

2194 `CELf_CS_MONITOR_LINE_STATUS`: Line status change notification

2195 `CELf_CS_MONITOR_RESTRICT`: Restriction status change notification

2196 `CELf_CS_MONITOR_RSSI`: Received signal strength change notification

2197 `CELf_CS_MONITOR_ALL`: All notified

2198

2199 28.1.3 Return Value

2200 Type: `CelfMpStatus`

2201 I/O: `O`

2202 Description:

2203 `celf_mp_cs_notification_stop()` **shall** return one of the values defined:

2204 `CELf_MP_STATUS_OK`: successful completion

2205 `CELf_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

2206 `CELf_MP_STATUS_MON_TYPE_ERR`: Monitor type is not valid

2207 `CELf_MP_STATUS_ERR`: Other unsuccessful completion.

2208

2209 28.1.4 Include File

2210 /usr/include/celf/mp_cs.h

2211

2212 28.1.5 Functional Description

2213 This function ends notifying on the event of the line status.

2214

DRAFT

2215 **29. Get Reception Level**

2216 **29.1 Symbol: celf_mp_cs_get_reception_level**

2217 **29.1.1 Syntax**

2218 CelfMpReceptionLevel celf_mp_cs_get_reception_level (
2219 void);

2220 **29.1.2 Argument**

2221 None.

2222

2223 **29.1.3 Return Value**

2224 Type: CelfMpReceptionLevel

2225 I/O: O

2226 Description:

2227 celf_mp_cs_get_reception_level() **shall** return one of the values defined:

2228 CELF_CS_RSSI_LEVEL_0: Receive level 0

2229 CELF_CS_RSSI_LEVEL_1: Receive level 1

2230 CELF_CS_RSSI_LEVEL_2: Receive level 2

2231 CELF_CS_RSSI_LEVEL_3: Receive level 3

2232

2233 **29.1.4 Include File**

2234 /usr/include/celf/mp_cs.h

2235

2236 **29.1.5 Functional Description**

2237 This function obtains the current reception level.

2238 Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the
2239 status of reception level.

2240

2241 **30. Get Line Status**

2242 **30.1 Symbol: celf_mp_cs_get_line_status**

2243 **30.1.1 Syntax**

2244 CelfMpStatus celf_mp_cs_get_line_status (
2245 CELF_CS_AREAREF_CHG_INF * net);

2246 **30.1.2 Argument**

2247 Name: net

2248 Type: CELF_CS_AREAREF_CHG_INF

2249 I/O: I

2250 Description:

2251 Pointer to the struct used to hold line status information

2252

2253 **30.1.3 Return Value**

2254 Type: CelfMpStatus

2255 I/O: O

2256 Description:

2257 celf_mp_cs_get_line_status() **shall** return one of the values defined:

2258 CELF_MP_STATUS_OK: successful completion

2259 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2260

2261 **30.1.4 Include File**

2262 /usr/include/celf/mp_cs.h

2263

2264 **30.1.5 Functional Description**

2265 This function obtains the current line status.

2266 Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the
2267 status of line status.

2268 See section 0.1 for further information.

2269

2270

31. Get Coverage Status

2271

31.1 Symbol: `celf_mp_cs_get_coverage_status`

2272

31.1.1 Syntax

2273

`CelfMpStatus celf_mp_cs_get_line_status (`

2274

`CELF_CS_LINE_STATUS_EX * net,`

2275

`CelfMpCsCoverage cover);`

2276

31.1.2 Argument

2277

Name: `net`

2278

Type: `CELF_CS_LINE_STATUS_EX`

2279

I/O: `I`

2280

Description:

2281

Pointer to the struct used to hold line status information

2282

2283

Name: `cover`

2284

Type: `CelfMpCsCoverage`

2285

I/O: `I`

2286

Description:

2287

Within- or out-of communication area status

2288

`CELF_CS_LINE_STATUS_IN`: Within-communication area

2289

`CELF_CS_LINE_STATUS_OUT`: Out-of-communication area

2290

2291

31.1.3 Return Value

2292

Type: `CelfMpStatus`

2293

I/O: `O`

2294

Description:

2295

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

2296

`CELF_MP_STATUS_OK`: successful completion

2297

`CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2298

2299

31.1.4 Include File

2300

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

2301

2302

31.1.5 Functional Description

2303

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

2304

for current line.

2305 (This function gets only information of inside or outside coverage area status.)
2306

DRAFT

2307 **32. Get Voice Mail Information**

2308 **32.1 Symbol: celf_mp_cs_get_vm_info**

2309 **32.1.1 Syntax**

2310 CelfMpStatus celf_mp_cs_get_vm_info (
2311 CelfMpCsVMNum * vm_num);

2312 **32.1.2 Argument**

2313 Name: vm_num

2314 Type: CelfMpCsVMNum

2315 I/O: I

2316 Description:

2317 Address of the storage area of the number of stored phone-answering messages

2318

2319 **32.1.3 Return Value**

2320 Type: CelfMpStatus

2321 I/O: O

2322 Description:

2323 celf_mp_cs_get_vm_info() **shall** return one of the values defined:

2324 CELF_MP_STATUS_OK: successful completion

2325 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2326

2327 **32.1.4 Include File**

2328 /usr/include/celf/mp_cs.h

2329

2330 **32.1.5 Functional Description**

2331 This function obtains the storage status of phone-answering messages from nonvolatile memory.

2332 The storage status is the number of message of phone-answering.

2333

2334

2335 **33.Set Voice Mail Information**

2336 **33.1 Symbol: celf_mp_cs_set_vm_info**

2337 **33.1.1 Syntax**

2338 CelfMpStatus celf_mp_cs_set_vm_info (
2339 CelfMpCsVMNum vm_num);

2340 **33.1.2 Argument**

2341 Name: vm_num

2342 Type: CelfMpCsVMNum

2343 I/O: I

2344 Description:

2345 The number of stored phone-answering messages

2346

2347 **33.1.3 Return Value**

2348 Type: CelfMpStatus

2349 I/O: O

2350 Description:

2351 celf_mp_cs_set_vm_info() **shall** return one of the values defined:

2352 CELF_MP_STATUS_OK: successful completion

2353 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2354

2355 **33.1.4 Include File**

2356 /usr/include/celf/mp_cs.h

2357

2358 **33.1.5 Functional Description**

2359 This function sets the storage status of phone-answering message to non-volatile memory.

2360

2361

2362

2363 **34. Get Call Selection**

2364 **34.1 Symbol: celf_mp_cs_get_call_select**

2365 **34.1.1 Syntax**

2366 CelfMpCallSelect celf_mp_cs_get_call_select (
2367 void);

2368 **34.1.2 Argument**

2369 None.

2370

2371 **34.1.3 Return Value**

2372 Type: CelfMpCallSelect

2373 I/O: O

2374 Description:

2375 CELF_CS_INCOMING_VOICE_ANSWERING: Forward to the phone-answering message

2376 CELF_CS_INCOMING_FORWARD: Forward

2377 CELF_CS_INCOMING_REJECT: Reject (disconnect)

2378 CELF_CS_INCOMING_NORMAL: Receipt of an incoming call (normal incoming)

2379

2380 **34.1.4 Include File**

2381 /usr/include/celf/mp_cs.h

2382

2383 **34.1.5 Functional Description**

2384 This function obtains the incoming call information from non-volatile memory.

2385 Refer "Set incoming function selection"

2386

2387

2388

35.Set Call Selection

2389

35.1 Symbol: `celf_mp_cs_set_call_select`

2390

35.1.1 Syntax

2391

`CelfMpStatus celf_mp_cs_set_call_select (`

2392

`CelfMpCallSelect select);`

2393

35.1.2 Argument

2394

Name: `select`

2395

Type: `CelfMpCallSelect`

2396

I/O: `I`

2397

Description:

2398

`CELf_CS_INCOMING_VOICE_ANSWERING:` Forward to the phone-answering message

2399

`CELf_CS_INCOMING_FORWARD:` Forward

2400

`CELf_CS_INCOMING_REJECT:` Reject (disconnect)

2401

`CELf_CS_INCOMING_NORMAL:` Receipt of an incoming call (normal incoming)

2402

2403

35.1.3 Return Value

2404

Type: `CelfMpStatus`

2405

I/O: `O`

2406

Description:

2407

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

2408

`CELf_MP_STATUS_OK:` successful completion

2409

`CELf_MP_STATUS_ERR:` Other unsuccessful completion.

2410

2411

35.1.4 Include File

2412

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

2413

2414

35.1.5 Functional Description

2415

This function sets the incoming call information to nonvolatile memory.

2416

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

2417

information.

2418

2419

2420

36.Set Service Information

2421

36.1 Symbol: `celf_mp_cs_set_service_info`

2422

36.1.1 Syntax

2423

```
CelfMpStatus celf_mp_cs_set_service_info (
```

2424

```
    CelfMpRegNum      reg_no,
```

2425

```
    CelfMpCsSrvData * data);
```

2426

36.1.2 Argument

2427

Name: `reg_no`

2428

Type: `CelfMpRegNum`

2429

I/O: `I`

2430

Description:

2431

Registration number: 1 to 10

2432

Name: `data`

2433

Type: `CelfMpCsSrvData`

2434

I/O: `I`

2435

Description:

2436

Pointer to supplementary service data

2437

2438

36.1.3 Return Value

2439

Type: `CelfMpStatus`

2440

I/O: `O`

2441

Description:

2442

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

2443

`CELf_MP_STATUS_OK`: successful completion

2444

`CELf_MP_STATUS_ERR`: Other unsuccessful completion.

2445

2446

36.1.4 Include File

2447

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

2448

2449

36.1.5 Functional Description

2450

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

2451

2452

The supplementary service information is the service name and Dial data for accessing the service.

2453

The `'reg_no'` is used as the key for accessing this supplementary service.

- 2454 The value range is from 0 to 10.
- 2455 See section 0.1 for additional information.
- 2456

DRAFT

2457 37. Get Service Information

2458 37.1 Symbol: `celf_mp_cs_get_service_info`

2459 37.1.1 Syntax

```
2460 CelfMpStatus celf_mp_cs_get_service_info (  
2461     CelfMpRegNum     reg_no,  
2462     CelfMpCsSrvData * data);
```

2463 37.1.2 Argument

2464 Name: `reg_no`

2465 Type: `CelfMpRegNum`

2466 I/O: `I`

2467 Description:

2468 Registration number: 1 to 10

2469 Name: `data`

2470 Type: `CelfMpCsSrvData`

2471 I/O: `I`

2472 Description:

2473 Pointer to supplementary service data

2474

2475 37.1.3 Return Value

2476 Type: `CelfMpStatus`

2477 I/O: `O`

2478 Description:

2479 `celf_mp_cs_get_service_info()` **shall** return one of the values defined:

2480 `CELF_MP_STATUS_OK`: successful completion

2481 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2482

2483 37.1.4 Include File

2484 `/usr/include/celf/mp_cs.h`

2485

2486 37.1.5 Functional Description

2487 This function obtains supplementary service information, specified by “`reg_no`”, from non-volatile
2488 memory.

2489

2490 See “Register supplementary service settings”.

2491 38.Delete Service Information

2492 38.1 Symbol: `celf_mp_cs_del_service_info`

2493 38.1.1 Syntax

2494 `CelfMpStatus celf_mp_cs_del_service_info (`
2495 `CelfMpRegNum reg_no);`

2496 38.1.2 Argument

2497 Name: `reg_no`

2498 Type: `CelfMpRegNum`

2499 I/O: `I`

2500 Description:

2501 Registration number: 1 to 10

2502

2503 38.1.3 Return Value

2504 Type: `CelfMpStatus`

2505 I/O: `O`

2506 Description:

2507 `celf_mp_cs_del_service_info()` shall return one of the values defined:

2508 `CELF_MP_STATUS_OK:` successful completion

2509 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2510

2511 38.1.4 Include File

2512 `/usr/include/celf/mp_cs.h`

2513

2514 38.1.5 Functional Description

2515 This function deletes the supplementary service information specified by “`reg_no`” from non-volatile
2516 memory.

2517 **39.Remove Service Information**

2518 **39.1 Symbol: celf_mp_cs_remove_all_service_info**

2519 **39.1.1 Syntax**

2520 CelfMpStatus celf_mp_cs_remove_all_service_info (
2521 void);

2522 **39.1.2 Argument**

2523 None.

2524

2525 **39.1.3 Return Value**

2526 Type: CelfMpStatus

2527 I/O: O

2528 Description:

2529 celf_mp_cs_remove_all_service_info() shall return one of the values defined:

2530 CELF_MP_STATUS_OK: successful completion

2531 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2532

2533 **39.1.4 Include File**

2534 /usr/include/celf/mp_cs.h

2535

2536 **39.1.5 Functional Description**

2537 This function deletes all the supplementary service information from non-volatile memory.

2538 40.Set Response Message Settings

2539 40.1 Symbol: `celf_mp_cs_set_resp_msg`

2540 40.1.1 Syntax

```
2541 CelfMpStatus celf_mp_cs_set_resp_msg (  
2542     CelfMpRegNum reg_no,  
2543     CelfMpCsSrvData * data);
```

2544 40.1.2 Argument

2545 Name: `reg_no`

2546 Type: `CelfMpRegNum`

2547 I/O: `I`

2548 Description:

2549 Registration number: 1 to 10

2550 Name: `data`

2551 Type: `CelfMpCsSrvData`

2552 I/O: `I`

2553 Description:

2554 Pointer to the additional response message setting data area

2555

2556 40.1.3 Return Value

2557 Type: `CelfMpStatus`

2558 I/O: `O`

2559 Description:

2560 `celf_mp_cs_set_resp_msg()` shall return one of the values defined:

2561 `CELFP_MP_STATUS_OK`: successful completion

2562 `CELFP_MP_STATUS_ERR`: Other unsuccessful completion.

2563

2564 40.1.4 Include File

2565 `/usr/include/celf/mp_cs.h`

2566

2567 40.1.5 Functional Description

2568 This function registers the supplementary response message information for the supplementary service to
2569 the non-volatile memory,

2570

2571 When a supplementary service is activated, and corresponding message from the network is received, this
2572 supplementary response message is sent to the network.

2573

2574 The supplementary response message information is the service name and Dial data, which is response
2575 message to send the network.

2576

2577 The dial data should be USSD.

2578

2579 The “reg_no” is used as the key for accessing this supplementary response message .

2580 The value range is from 0 to 10.

2581

2582 For information about the structures, see section 0.1.

2583

DRAFT

2584

41. Get Response Message Settings

2585

41.1 Symbol: `celf_mp_cs_get_resp_msg`

2586

41.1.1 Syntax

2587

```
CelfMpStatus celf_mp_cs_get_resp_msg (
```

2588

```
    CelfMpRegNum reg_no,
```

2589

```
    CelfMpCsSrvData * data);
```

2590

41.1.2 Argument

2591

Name: `reg_no`

2592

Type: `CelfMpRegNum`

2593

I/O: `I`

2594

Description:

2595

Registration number: 1 to 10

2596

Name: `data`

2597

Type: `CelfMpCsSrvData`

2598

I/O: `I`

2599

Description:

2600

Pointer to the additional response message setting data area

2601

2602

41.1.3 Return Value

2603

Type: `CelfMpStatus`

2604

I/O: `O`

2605

Description:

2606

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

2607

`CELF_MP_STATUS_OK`: successful completion

2608

`CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2609

2610

41.1.4 Include File

2611

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

2612

2613

41.1.5 Functional Description

2614

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

2615

volatile memory.

2616

2617

See “Register response message settings”.

2618 42.Delete Response Message Settings

2619 42.1 Symbol: `celf_mp_cs_del_resp_msg`

2620 42.1.1 Syntax

```
2621 CelfMpStatus celf_mp_cs_del_resp_msg (  
2622     CelfMpRegNum reg_no);
```

2623 42.1.2 Argument

2624 Name: `reg_no`

2625 Type: `CelfMpRegNum`

2626 I/O: `I`

2627 Description:

2628 Registration number: 1 to 10

2629

2630 42.1.3 Return Value

2631 Type: `CelfMpStatus`

2632 I/O: `O`

2633 Description:

2634 `celf_mp_cs_del_resp_msg()` shall return one of the values defined:

2635 `CELF_MP_STATUS_OK`: successful completion

2636 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2637

2638 42.1.4 Include File

2639 `/usr/include/celf/mp_cs.h`

2640

2641 42.1.5 Functional Description

2642 This function deletes the supplementary response message information, specified by “`reg_no`”, from non-
2643 volatile memory.

2644 **43.Remove All Response Message Settings**

2645 **43.1 Symbol: celf_mp_cs_remove_all_resp_msg**

2646 **43.1.1 Syntax**

2647 CelfMpStatus celf_mp_cs_remove_all_resp_msg (
2648 void);

2649 **43.1.2 Argument**

2650 None.

2651

2652 **43.1.3 Return Value**

2653 Type: CelfMpStatus

2654 I/O: O

2655 Description:

2656 celf_mp_cs_remove_all_resp_msg() **shall** return one of the values defined:

2657 CELF_MP_STATUS_OK: successful completion

2658 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2659

2660 **43.1.4 Include File**

2661 /usr/include/celf/mp_cs.h

2662

2663 **43.1.5 Functional Description**

2664 This function removes from non-volatile memory all the supplementary response message information.

2665

2666 44.Set Reconnection Tone

2667 44.1 Symbol: `celf_mp_cs_set_reconnection_tone`

2668 44.1.1 Syntax

```
2669 CelfMpStatus celf_mp_cs_set_reconnection_tone (  
2670         CelfMpCsReconnectionTone     reconn);
```

2671 44.1.2 Argument

2672 Name: `reconn`

2673 Type: `CelfMpCsReconnectionTone`

2674 I/O: `I`

2675 Description:

2676 Reconnection tone to be set

2677 `CELF_CS_RECONN_ON_T_OFF:` Tone OFF

2678 `CELF_CS_RECONN_ON_T_LOW:` Tone ON low tone

2679 `CELF_CS_RECONN_ON_T_HI:` Tone ON high tone

2680

2681 44.1.3 Return Value

2682 Type: `CelfMpStatus`

2683 I/O: `O`

2684 Description:

2685 `celf_mp_cs_set_reconnection_tone()` shall return one of the values defined:

2686 `CELF_MP_STATUS_OK:` successful completion

2687 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2688

2689 44.1.4 Include File

2690 `/usr/include/celf/mp_cs.h`

2691

2692 44.1.5 Functional Description

2693 This function sets the reconnection tone information to the non-volatile memory.

2694 The type of reconnection tone is specified by “reconn”

2695

2696 45. Get Reconnection Tone

2697 45.1 Symbol: `celf_mp_cs_get_reconnection_tone`

2698 45.1.1 Syntax

```
2699 CelfMpCsReconnectionTone celf_mp_cs_get_reconnection_tone (  
2700     void);
```

2701 45.1.2 Argument

2702 None.

2703

2704 45.1.3 Return Value

2705 Type: `CelfMpCsReconnectionTone`

2706 I/O: `O`

2707 Description:

2708 `celf_mp_cs_get_reconnection_tone()` **shall** return one of the values defined:

2709 `CELF_CS_RECONN_ON_T_OFF`: Tone OFF

2710 `CELF_CS_RECONN_ON_T_LOW`: Tone ON low tone

2711 `CELF_CS_RECONN_ON_T_HI`: Tone ON high tone

2712

2713

2714 45.1.4 Include File

2715 `/usr/include/celf/mp_cs.h`

2716

2717 45.1.5 Functional Description

2718 This function gets the reconnection tone information to the non-volatile memory.

2719 **46.Get Noise Cancel**

2720 **46.1 Symbol: celf_mp_cs_get_noise_cancel**

2721 **46.1.1 Syntax**

2722 CelfMpCsNoiseCancel celf_mp_cs_get_noise_cancel (
2723 void);

2724 **46.1.2 Argument**

2725 None.

2726

2727 **46.1.3 Return Value**

2728 Type: CelfMpCsNoiseCancel

2729 I/O: O

2730 Description:

2731 celf_mp_cs_get_noise_cancel() **shall** return one of the values defined:

2732 CELF_CS_ON: Noise canceller ON

2733 CELF_CS_OFF: Noise canceller OFF

2734

2735 **46.1.4 Include File**

2736 /usr/include/celf/mp_cs.h

2737

2738 **46.1.5 Functional Description**

2739 This function gets the noise canceller status.

2740

2741 **47.Set Noise Cancel**

2742 **47.1 Symbol: celf_mp_cs_set_noise_cancel**

2743 **47.1.1 Syntax**

2744 CelfMpStatus celf_mp_cs_set_noise_cancel (
2745 CelfMpCsNoiseCancel mode);

2746 **47.1.2 Argument**

2747 Name: mode

2748 Type: CelfMpCsNoiseCancel

2749 I/O: I

2750 Description:

2751 Reconnection tone to be set

2752 CELF_CS_ON: Noise canceller ON

2753 CELF_CS_OFF: Noise canceller OFF

2754

2755 **47.1.3 Return Value**

2756 Type: CelfMpStatus

2757 I/O: O

2758 Description:

2759 `celf_mp_cs_set_noise_cancel()` **shall** return one of the values defined:

2760 CELF_MP_STATUS_OK: successful completion

2761 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2762

2763 **47.1.4 Include File**

2764 `/usr/include/celf/mp_cs.h`

2765

2766 **47.1.5 Functional Description**

2767 This function sets the noise canceller off or on.

2768 **48.Get Quality Alarm**

2769 **48.1 Symbol: celf_mp_cs_get_quality_alarm**

2770 **48.1.1 Syntax**

2771 CelfMpCsQualAlarm celf_mp_cs_get_quality_alarm(
2772 void);

2773 **48.1.2 Argument**

2774 None.

2775

2776 **48.1.3 Return Value**

2777 Type: CelfMpCsQualAlarm

2778 I/O: O

2779 Description:

2780 celf_mp_cs_get_quality_alarm() **shall** return one of the values defined:

2781 CELF_CS_QUALITY_ALM_OFF: Quality alarm OFF

2782 CELF_CS_QUALITY_ALM_LOW: Quality alarm ON low tone

2783 CELF_CS_QUALITY_ALM_HI: Quality alarm ON high tone

2784

2785 **48.1.4 Include File**

2786 /usr/include/celf/mp_cs.h

2787

2788 **48.1.5 Functional Description**

2789 This function gets the status of the call quality alarm sound.

2790 49.Set Quality Alarm

2791 49.1 Symbol: `celf_mp_cs_set_quality_alarm`

2792 49.1.1 Syntax

```
2793 CelfMpStatus celf_mp_cs_set_quality_alarm (  
2794     CelfMpCsQualAlarm mode);
```

2795 49.1.2 Argument

2796 Name: mode

2797 Type: CelfMpCsQualAlarm

2798 I/O: I

2799 Description:

2800 CELF_CS_QUALITY_ALM_OFF: Quality alarm OFF

2801 CELF_CS_QUALITY_ALM_LOW: Quality alarm ON low tone

2802 CELF_CS_QUALITY_ALM_HI: Quality alarm ON high tone

2803

2804 49.1.3 Return Value

2805 Type: CelfMpStatus

2806 I/O: O

2807 Description:

2808 `celf_mp_cs_set_quality_alarm()` shall return one of the values defined:

2809 CELF_MP_STATUS_OK: successful completion

2810 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2811

2812 49.1.4 Include File

2813 `/usr/include/celf/mp_cs.h`

2814

2815 49.1.5 Functional Description

2816 This function sets the call quality alarm sound.

2817 **50.Get Noise Cancel Permit**

2818 **50.1 Symbol: celf_mp_cs_get_noise_cancel_permit**

2819 **50.1.1 Syntax**

2820 CelfMpCsNoiseCancel celf_mp_cs_get_noise_cancel_permit(
2821 void);

2822 **50.1.2 Argument**

2823 None.

2824

2825 **50.1.3 Return Value**

2826 Type: CelfMpCsNoiseCancel

2827 I/O: O

2828 Description:

2829 celf_mp_cs_get_noise_cancel_permit() shall return one of the values defined:

2830 CELF_CS_ON: Noise canceller permission

2831 CELF_CS_OFF: Noise canceller non-permission

2832

2833 **50.1.4 Include File**

2834 /usr/include/celf/mp_cs.h

2835

2836 **50.1.5 Functional Description**

2837 This function obtains whether noise canceller is permitted or not.

2838 **51.Set High Priority communication mode**

2839 **51.1 Symbol: celf_mp_cs_set_hi_prio_com**

2840 **51.1.1 Syntax**

2841 CelfMpStatus celf_mp_cs_set_hi_prio_com (
2842 CelfMpCsHiPrioCom mode);

2843 **51.1.2 Argument**

2844 Name: mode

2845 Type: CelfMpCsHiPrioCom

2846 I/O: I

2847 Description:

2848 Reconnection tone to be set

2849 CELF_CS_COMPRI_NONE: No setting

2850 CELF_CS_COMPRI_VOICE: Voice

2851 CELF_CS_COMPRI_PACKET: Packet

2852

2853 **51.1.3 Return Value**

2854 Type: CelfMpStatus

2855 I/O: O

2856 Description:

2857 `celf_mp_cs_set_hi_prio_com()` shall return one of the values defined:

2858 CELF_MP_STATUS_OK: successful completion

2859 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2860

2861 **51.1.4 Include File**

2862 `/usr/include/celf/mp_cs.h`

2863

2864 **51.1.5 Functional Description**

2865 This function sets the high priority communication mode either on the voice communication or on the
2866 packet communication.

2867 **52. Get Phone Answering Sound Activation**

2868 **52.1 Symbol: celf_mp_cs_get_vm_sound_status**

2869 **52.1.1 Syntax**

2870 CelfMpCsVmSound celf_mp_cs_get_vm_sound_status(
2871 void);

2872 **52.1.2 Argument**

2873 None.

2874

2875 **52.1.3 Return Value**

2876 Type: CelfMpCsVmSound

2877 I/O: O

2878 Description:

2879 celf_mp_cs_get_vm_sound_status() **shall** return one of the values defined:

2880 CELF_CS_ON: Message sound ON

2881 CELF_CS_OFF: Message sound OFF

2882

2883 **52.1.4 Include File**

2884 /usr/include/celf/mp_cs.h

2885

2886 **52.1.5 Functional Description**

2887 This function gets the setting status.

2888 IF the setting status is ON, the phone sounds, when the number of voice mail system is increased.

2889 **53.Set Phone Answering Sound Activation**

2890 **53.1 Symbol: celf_mp_cs_set_vm_sound_status**

2891 **53.1.1 Syntax**

```
2892 CelfMpStatus celf_mp_cs_get_vm_sound_status (  
2893     CelfMpCsVmSound mode);
```

2894 **53.1.2 Argument**

2895 Type: CelfMpCsVmSound

2896 I/O: O

2897 Description:

2898 CELF_CS_ON: Message sound ON

2899 CELF_CS_OFF: Message sound OFF

2900

2901 **53.1.3 Return Value**

2902 Type: CelfMpStatus

2903 I/O: O

2904 Description:

2905 `celf_mp_cs_set_vm_sound_status()` shall return one of the values defined:

2906 CELF_MP_STATUS_OK: successful completion

2907 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2908

2909 **53.1.4 Include File**

2910 `/usr/include/celf/mp_cs.h`

2911

2912 **53.1.5 Functional Description**

2913 This function sets the phone sounds status whether the phone sounds or not.

2914 **54. Get Automatic Receive Status**

2915 **54.1 Symbol: celf_mp_cs_get_auto_rcv_status**

2916 **54.1.1 Syntax**

2917 CelfMpCsVmSound celf_mp_cs_get_auto_rcv_status(
2918 void);

2919 **54.1.2 Argument**

2920 None.

2921

2922 **54.1.3 Return Value**

2923 Type: CelfMpCsVmSound

2924 I/O: O

2925 Description:

2926 celf_mp_cs_get_auto_rcv_status() **shall** return one of the values defined:

2927 CELF_CS_ON: Automatic incoming call ON

2928 CELF_CS_OFF: Automatic incoming call OFF

2929

2930 **54.1.4 Include File**

2931 /usr/include/celf/mp_cs.h

2932

2933 **54.1.5 Functional Description**

2934 This function obtains the status of automatic incoming call.

2935 The status is ON or OFF.

2936 **55.Set Automatic Receive Status**

2937 **55.1 Symbol: celf_mp_cs_set_auto_rcv_status**

2938 **55.1.1 Syntax**

2939 CelfMpStatus celf_mp_cs_set_auto_rcv_status (
2940 CelfMpCsAutoRcv mode);

2941 **55.1.2 Argument**

2942 Type: CelfMpCsAutoRcv

2943 I/O: O

2944 Description:

2945 CELF_CS_ON: Automatic incoming call ON

2946 CELF_CS_OFF: Automatic incoming call OFF

2947

2948 **55.1.3 Return Value**

2949 Type: CelfMpStatus

2950 I/O: O

2951 Description:

2952 celf_mp_cs_set_auto_rcv_status() **shall** return one of the values defined:

2953 CELF_MP_STATUS_OK: successful completion

2954 CELF_MP_STATUS_ERR: Other unsuccessful completion.

2955

2956 **55.1.4 Include File**

2957 /usr/include/celf/mp_cs.h

2958

2959 **55.1.5 Functional Description**

2960 This function sets the automatic incoming call status.

2961 **56.Get Automatic Timer**

2962 **56.1 Symbol: celf_mp_cs_get_auto_timer**

2963 **56.1.1 Syntax**

2964 CelfMpCsTimer celf_mp_cs_get_auto_timer(
2965 void);

2966 **56.1.2 Argument**

2967 None.

2968

2969 **56.1.3 Return Value**

2970 Type: CelfMpCsTimer

2971 I/O: O

2972 Description:

2973 `celf_mp_cs_get_auto_timer()` **shall** return one of the values defined:

2974 1 to 120 seconds

2975

2976 **56.1.4 Include File**

2977 `/usr/include/celf/mp_cs.h`

2978

2979 **56.1.5 Functional Description**

2980 This function obtains the timer value of the automatic incoming call.

2981 The timer value is the duration of sounding of the ring alert.

2982 **57.Set Automatic Timer**

2983 **57.1 Symbol: celf_mp_cs_set_auto_timer**

2984 **57.1.1 Syntax**

2985 CelfMpStatus celf_mp_cs_set_auto_timer (
2986 CelfMpCsTimer time);

2987 **57.1.2 Argument**

2988 Type: CelfMpCsTimer

2989 I/O: O

2990 Description:

2991 1 to 120 seconds

2992

2993 **57.1.3 Return Value**

2994 Type: CelfMpStatus

2995 I/O: O

2996 Description:

2997 celf_mp_cs_set_auto_timer() **shall** return one of the values defined:

2998 CELF_MP_STATUS_OK: successful completion

2999 CELF_MP_STATUS_ERR: Other unsuccessful completion.

3000

3001 **57.1.4 Include File**

3002 /usr/include/celf/mp_cs.h

3003

3004 **57.1.5 Functional Description**

3005 This function sets the timer value of the automatic incoming call.

3006

3007

3008

3009

3010 **58. Get Reset Date**

3011 **58.1 Symbol: celf_mp_cs_get_reset_date**

3012 **58.1.1 Syntax**

3013 CelfMpStatus celf_mp_cs_get_reset_date(
3014 CelfMpCsDate * reset_date);

3015 **58.1.2 Argument**

3016 Type: CelfMpCsDate

3017 I/O: O

3018 Description:

3019 Accumulated date record

3020 See section 0.1 for details.

3021

3022

3023 **58.1.3 Return Value**

3024 Type: CelfMpStatus

3025 I/O: O

3026 Description:

3027 `celf_mp_cs_get_reset_date()` shall return one of the values defined:

3028 CELF_MP_STATUS_OK: successful completion

3029 CELF_MP_STATUS_ERR: Other unsuccessful completion.

3030

3031 **58.1.4 Include File**

3032 `/usr/include/celf/mp_cs.h`

3033

3034 **58.1.5 Functional Description**

3035 This function obtains the date and time when the accumulated call duration was reset.

3036 The value is obtained from non-volatile memory.

3037 **59.Set Reset Date**

3038 **59.1 Symbol: celf_mp_cs_set_reset_date**

3039 **59.1.1 Syntax**

3040 CelfMpStatus celf_mp_cs_set_reset_date(
3041 void);

3042 **59.1.2 Argument**

3043 None.

3044

3045 **59.1.3 Return Value**

3046 Type: CelfMpStatus

3047 I/O: O

3048 Description:

3049 celf_mp_cs_set_reset_date() **shall** return one of the values defined:

3050 CELF_MP_STATUS_OK: successful completion

3051 CELF_MP_STATUS_ERR: Other unsuccessful completion.

3052

3053 **59.1.4 Include File**

3054 /usr/include/celf/mp_cs.h

3055

3056 **59.1.5 Functional Description**

3057 This function sets the current date and time as the reset date and time of the accumulated call duration.

3058 The value set to non-volatile memory.

3059 **60.Get Call Start Time**

3060 **60.1 Symbol: celf_mp_cs_get_call_start_time**

3061 **60.1.1 Syntax**

3062 CelfMpTime celf_mp_cs_get_call_start_time(
3063 void);

3064 **60.1.2 Argument**

3065 None.

3066

3067

3068 **60.1.3 Return Value**

3069 Type: CelfMpTime

3070 I/O: O

3071 Description:

3072 celf_mp_cs_get_call_start_time() **shall** return one of the values defined:

3073 0 to 99 seconds

3074

3075 **60.1.4 Include File**

3076 /usr/include/celf/mp_cs.h

3077

3078 **60.1.5 Functional Description**

3079 This function gets the duration between the arrival of incoming call and the start of sounding of the ring
3080 alert. This duration is called the silent time.

3081 This function is effective that the number of this incoming call is unregistered with the phone book.

3082 **61.Set Call Start Time**

3083 **61.1 Symbol: celf_mp_cs_set_call_start_time**

3084 **61.1.1 Syntax**

3085 CelfMpStatus celf_mp_cs_set_call_start_time(
3086 CelfMpCsTimer time);

3087 **61.1.2 Argument**

3088 Type: CelfMpCsTimer

3089 I/O: O

3090 Description:

3091 0 to 99 seconds

3092

3093 **61.1.3 Return Value**

3094 Type: CelfMpStatus

3095 I/O: O

3096 Description:

3097 celf_mp_cs_set_call_start_time() **shall** return one of the values defined:

3098 CELF_MP_STATUS_OK: successful completion

3099 CELF_MP_STATUS_ERR: Other unsuccessful completion.

3100

3101 **61.1.4 Include File**

3102 /usr/include/celf/mp_cs.h

3103

3104 **61.1.5 Functional Description**

3105 This function sets the silent time.

3106 Refer to get calling operation start time.

3107

62.Get Call Recorded

3108

62.1 Symbol: `celf_mp_cs_get_call_recorded`

3109

62.1.1 Syntax

3110

`CelfMpSetting celf_mp_cs_get_call_recorded(`

3111

`void);`

3112

62.1.2 Argument

3113

None.

3114

3115

3116

62.1.3 Return Value

3117

Type: `CelfMpSetting`

3118

I/O: `O`

3119

Description:

3120

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

3121

`CELF_CS_ON`: Setting ON

3122

`CELF_CS_OFF`: Setting OFF

3123

3124

62.1.4 Include File

3125

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

3126

3127

62.1.5 Functional Description

3128

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

3129

3130

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

3131

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

3132

Refer to “Get calling operation start time”.

3133

3134

3135

3136 63.Set Call Recorded

3137 63.1 Symbol: `celf_mp_cs_set_call_recorded`

3138 63.1.1 Syntax

3139 `CelfMpStatus celf_mp_cs_set_call_recorded(
3140 CelfMpCsSetting mode);`

3141 63.1.2 Argument

3142 Type: `CelfMpCsSetting`

3143 I/O: `O`

3144 Description:

3145 `CELF_CS_ON`: Setting ON

3146 `CELF_CS_OFF`: Setting OFF

3147

3148 63.1.3 Return Value

3149 Type: `CelfMpStatus`

3150 I/O: `O`

3151 Description:

3152 `celf_mp_cs_set_call_start_time()` shall return one of the values defined:

3153 `CELF_MP_STATUS_OK`: successful completion

3154 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

3155

3156 63.1.4 Include File

3157 `/usr/include/celf/mp_cs.h`

3158

3159 63.1.5 Functional Description

3160 This function sets the setting condition of whether the silent call is recorded in the absent incoming call log,
3161 or not.

3162 Refer to “Get recording condition to absent incoming call log”.

3163

3164

DRAFT