



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  
27

## **Linux based 3G Specification**

### **Multimedia Mobile Phone API**

#### ***Circuit Switched Communication Service***

Document:        CELF\_MPP\_CS\_FR2b\_20060611

**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**

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
2.2.4	Review changes		AK	06/06/08
2.2.5	Clarifications	NEC	AK	06/06/11

DRAFT

29

30	<b>0. Introduction.....</b>	<b>15</b>
31	<b>References.....</b>	<b>16</b>
32	0.1.1 Normative .....	16
33	0.1.2 Informative .....	16
34	<b>1. Primitives.....</b>	<b>17</b>
35	<b>1.1 Constants.....</b>	<b>17</b>
36	1.1.1 Line type .....	17
37	1.1.2 Dial Number .....	17
38	1.1.3 TAF address.....	17
39	1.1.4 Supplementary service.....	17
40	<b>1.2 Enums.....</b>	<b>18</b>
41	1.2.1 Voice communication status (CelfMpCsComStatus).....	18
42	1.2.2 Forwarding result (CelfMpCSFwResult).....	19
43	1.2.3 Forwarding result details (CelfMpCsFwError).....	19
44	1.2.4 Communication type (CelfMpCsBtype).....	19
45	1.2.5 Call Reference Status.....	19
46	1.2.6 Call Status.....	19
47	1.2.7 Existence of continuation data.....	20
48	1.2.8 Busy Tone sound flag.....	20
49	1.2.9 Cause of NoCLI.....	20
50	1.2.10 Dial number / Redirect number display indicator.....	21
51	1.2.11 Signal information (CelfMpCsSignal).....	21
52	1.2.12 Originating Number notification (CelfMpCsNotice).....	21
53	1.2.13 Line status (CelfMpCsLineStatus).....	21
54	1.2.14 Normal and emergency originating restriction .....	21
55	1.2.15 Receive level (CelfMpCsRSSILevel).....	21
56	1.2.16 Area status information (CelfMpCsLineCvrStatus).....	22
57	1.2.17 RRC mode (CelfMpCsLineRRCMode).....	22
58	1.2.18 Service status (CelfMpLineSrvStatus).....	22
59	1.2.19 Restriction status (CelfMpCsLineRestrict).....	22
60	1.2.20 Identifying flag (CelfMpCsFlag).....	22
61	1.2.21 Notification Set (CelfMpCsNotifySet).....	22
62	1.2.22 Event Structure Category.....	23
63	1.2.23 Event Structure Subtype.....	23
64	1.2.24 Call Number (CelfMpCallNo).....	23
65	1.2.25 DCF Event Set (CelfMpCsDCFSet).....	23
66	1.2.26 Voice message (CelfMpCsRecMsg).....	23
67	1.2.27 Off Hook Option (CelfMpCsOffHk).....	24
68	1.2.28 64K/AV Communication (CelfMpCsUDComStatus).....	24
69	1.2.29 AV Communication (CelfMpAVComStatus).....	24
70	1.2.30 Receive Scene Events (CelfMpCsRcvScene).....	24
71	1.2.31 Line Monitoring (CelfMpCsMtype).....	25
72	1.2.32 Reception Level (CelfMpReceptionLevel).....	25
73	1.2.33 Coverage Indicators (CelfMpCsCoverage).....	25
74	1.2.34 Incoming Call Selection (CelfMpCallSelect).....	25
75	1.2.35 Registration number (CelfMpRegNum).....	25
76	1.2.36 Service Data (CelfMpCsSrvData).....	25
77	1.2.37 Reconnection Tone (CelfMpCsReconnectionTone).....	25
78	1.2.38 Noise Canceling (CelfMpCsNoiseCancel).....	26
79	1.2.39 Quality Alarm (CelfMpCsQualAlarm).....	26
80	1.2.40 Reconnection Tone Priority (CelfMpCsHiPrioCom).....	26

Classification: *Circuit Switched Service*

81	1.2.41	Message Sound settings (CelfMpCsVmSound).....	26
82	1.2.42	Incoming Call Auto Receive (CelfMpCsAutoRcv) .....	26
83	<b>1.3</b>	<b>Data Types and Structures .....</b>	<b>27</b>
84	1.3.1	Circuit switched status notification event structure .....	27
85	1.3.2	Call duration notification event structure.....	27
86	1.3.3	Disconnection cause notification event structure.....	27
87	1.3.4	Disconnection cause information structure.....	27
88	1.3.5	Forwarding result notification event structure .....	28
89	1.3.6	Forwarding result structure (CelfMpCsFwResult).....	28
90	1.3.7	Off-hook transmission timeout event structure.....	28
91	1.3.8	Connection Destination Information (CelfMpConnectInfo).....	28
92	1.3.9	Connection Request (CelfMpCsConReq).....	29
93	1.3.10	Redirection number.....	29
94	1.3.11	Channel Number Information (CelfMpCsChanNum).....	29
95	1.3.12	Channel not in use Flag .....	29
96	1.3.13	DCF Event Structure.....	29
97	1.3.14	Line status change notification event structure .....	30
98	1.3.15	Restriction display information structure (CelfMpCsResChgInf) .....	30
99	1.3.16	Receive level change notification event structure .....	30
100	1.3.17	Line Status structure (CelfMpCsAreaRefChgInf) .....	30
101	1.3.18	Supplementary service data structure (CelfMpCsAddsrvData).....	31
102	1.3.19	Response Message Data Structure (CelfMpCsResponseMsgData) .....	31
103	1.3.20	Line Status Extension (CelfMpCsLineStatusEx).....	31
104	1.3.21	Number of stored messages (CelfMpCsVMNum).....	31
105	1.3.22	Date Format Structure (CelfMpCsDate).....	31
106	1.3.23	Dial Buffer (CelfMpCsDialBuffer).....	32
107	1.3.24	Dial Buffer Length (CelfMpCsDialLen).....	32
108	1.3.25	Multi Party Operation (CelfMpCsMop).....	32
109	1.3.26	Timer Value (CelfMpCsTimer).....	32
110	<b>1.4</b>	<b>Events Type.....</b>	<b>33</b>
111	1.4.1	DCF Event Type .....	33
112	1.4.2	CCP Notification type.....	33
113	1.4.3	Notification type .....	34
114	1.4.4	Restriction status.....	34
115	<b>2.</b>	<b>Start Notification.....</b>	<b>35</b>
116	<b>2.1</b>	<b>Symbol: celf_mp_cs_notification_start.....</b>	<b>35</b>
117	2.1.1	Syntax .....	35
118	2.1.2	Argument .....	35
119	2.1.3	Return Value.....	36
120	2.1.4	Include File.....	36
121	2.1.5	Functional Description.....	36
122	<b>3.</b>	<b>Stop Notification .....</b>	<b>37</b>
123	<b>3.1</b>	<b>Symbol: celf_mp_cs_notification_stop .....</b>	<b>37</b>
124	3.1.1	Syntax .....	37
125	3.1.2	Argument .....	37
126	3.1.3	Return Value.....	37
127	3.1.4	Include File.....	38
128	3.1.5	Functional Description.....	38
129	<b>4.</b>	<b>Get Voice Communication Status .....</b>	<b>39</b>
130	<b>4.1</b>	<b>Symbol: celf_mp_cs_get_com_status.....</b>	<b>39</b>

131	4.1.1	Syntax .....	39
132	4.1.2	Argument .....	39
133	4.1.3	Return Value .....	39
134	4.1.4	Include File .....	39
135	4.1.5	Functional Description.....	39
136	<b>5.</b>	<b><i>Get Connection Information to Other Party</i></b> .....	<b>40</b>
137	<b>5.1</b>	<b>Symbol: <i>celf_mp_cs_get_con_info_ref</i></b> .....	<b>40</b>
138	5.1.1	Syntax .....	40
139	5.1.2	Argument .....	40
140	5.1.3	Return Value .....	40
141	5.1.4	Include File .....	41
142	5.1.5	Functional Description.....	41
143	<b>6.</b>	<b><i>Get Call Duration</i></b> .....	<b>42</b>
144	<b>6.1</b>	<b>Symbol: <i>celf_mp_cs_get_call_duration</i></b> .....	<b>42</b>
145	6.1.1	Syntax .....	42
146	6.1.2	Argument .....	42
147	6.1.3	Return Value .....	42
148	6.1.4	Include File .....	42
149	6.1.5	Functional Description.....	42
150	<b>7.</b>	<b><i>Off-Hook Notification</i></b> .....	<b>44</b>
151	<b>7.1</b>	<b>Symbol: <i>celf_mp_cs_notification_off_hook</i></b> .....	<b>44</b>
152	7.1.1	Syntax .....	44
153	7.1.2	Argument .....	44
154	7.1.3	Return Value .....	44
155	7.1.4	Include File .....	45
156	7.1.5	Functional Description.....	45
157	<b>8.</b>	<b><i>Disconnect</i></b> .....	<b>46</b>
158	<b>8.1</b>	<b>Symbol: <i>celf_mp_cs_disconnect</i></b> .....	<b>46</b>
159	8.1.1	Syntax .....	46
160	8.1.2	Argument .....	46
161	8.1.3	Return Value .....	46
162	8.1.4	Include File .....	46
163	8.1.5	Functional Description.....	46
164	<b>9.</b>	<b><i>Dial</i></b> .....	<b>48</b>
165	<b>9.1</b>	<b>Symbol: <i>celf_mp_cs_dial</i></b> .....	<b>48</b>
166	9.1.1	Syntax .....	48
167	9.1.2	Argument .....	48
168	9.1.3	Return Value .....	48
169	9.1.4	Include File .....	49
170	9.1.5	Functional Description.....	49
171	<b>10.</b>	<b><i>Dial Complete</i></b> .....	<b>50</b>
172	<b>10.1</b>	<b>Symbol: <i>celf_mp_cs_dial_end</i></b> .....	<b>50</b>
173	10.1.1	Syntax .....	50
174	10.1.2	Argument .....	50
175	10.1.3	Return Value .....	50
176	10.1.4	Include File .....	50
177	10.1.5	Functional Description.....	50

178	<b>11. Response to Incoming Call</b> .....	<b>52</b>
179	<b>11.1 Symbol: celf_mp_cs_call_rcv</b> .....	<b>52</b>
180	11.1.1 Syntax .....	52
181	11.1.2 Argument .....	52
182	11.1.3 Return Value.....	52
183	11.1.4 Include File .....	52
184	11.1.5 Functional Description.....	52
185	<b>12. Forward Incoming Call</b> .....	<b>54</b>
186	<b>12.1 Symbol: celf_mp_cs_call_forward</b> .....	<b>54</b>
187	12.1.1 Syntax .....	54
188	12.1.2 Argument .....	54
189	12.1.3 Return Value.....	54
190	12.1.4 Include File .....	54
191	12.1.5 Functional Description.....	54
192	<b>13. Forward to Voice Mail System</b> .....	<b>56</b>
193	<b>13.1 Symbol: celf_mp_cs_call_forward_voice_msg</b> .....	<b>56</b>
194	13.1.1 Syntax .....	56
195	13.1.2 Argument .....	56
196	13.1.3 Return Value.....	56
197	13.1.4 Include File .....	56
198	13.1.5 Functional Description.....	56
199	<b>14. Call Hold</b> .....	<b>57</b>
200	<b>14.1 Symbol: celf_mp_cs_call_hold</b> .....	<b>57</b>
201	14.1.1 Syntax .....	57
202	14.1.2 Argument .....	57
203	14.1.3 Return Value.....	57
204	14.1.4 Include File .....	57
205	14.1.5 Functional Description.....	57
206	<b>15. Call Reject</b> .....	<b>59</b>
207	<b>15.1 Symbol: celf_mp_cs_call_reject</b> .....	<b>59</b>
208	15.1.1 Syntax .....	59
209	15.1.2 Argument .....	59
210	15.1.3 Return Value.....	59
211	15.1.4 Include File .....	59
212	15.1.5 Functional Description.....	59
213	<b>16. Multi Party Call</b> .....	<b>61</b>
214	<b>16.1 Symbol: celf_mp_cs_mp_call</b> .....	<b>61</b>
215	16.1.1 Syntax .....	61
216	16.1.2 Argument .....	61
217	16.1.3 Return Value.....	61
218	16.1.4 Include File .....	62
219	16.1.5 Functional Description.....	62
220	<b>17. On-Hook Originating</b> .....	<b>64</b>
221	<b>17.1 Symbol: celf_mp_cs_originating_on_hook</b> .....	<b>64</b>
222	17.1.1 Syntax .....	64
223	17.1.2 Argument .....	64

224	17.1.3	Return Value.....	64
225	17.1.4	Include File.....	64
226	17.1.5	Functional Description.....	65
227	<b>18.</b>	<b><i>Get Call Reference</i></b> .....	<b>66</b>
228	<b>18.1</b>	<b>Symbol: <i>celf_mp_cs_get_call_reference</i></b> .....	<b>66</b>
229	18.1.1	Syntax .....	66
230	18.1.2	Argument .....	66
231	18.1.3	Return Value.....	66
232	18.1.4	Include File.....	66
233	18.1.5	Functional Description.....	66
234	<b>19.</b>	<b><i>Start DCF message notification</i></b> .....	<b>68</b>
235	<b>19.1</b>	<b>Symbol: <i>celf_mp_cs_DCF_notification_start</i></b> .....	<b>68</b>
236	19.1.1	Syntax .....	68
237	19.1.2	Argument .....	68
238	19.1.3	Return Value.....	69
239	19.1.4	Include File.....	69
240	19.1.5	Functional Description.....	69
241	<b>20.</b>	<b><i>Stop DCF message notification</i></b> .....	<b>71</b>
242	<b>20.1</b>	<b>Symbol: <i>celf_mp_cs_DCF_notification_stop</i></b> .....	<b>71</b>
243	20.1.1	Syntax .....	71
244	20.1.2	Argument .....	71
245	20.1.3	Return Value.....	71
246	20.1.4	Include File.....	72
247	20.1.5	Functional Description.....	72
248	<b>21.</b>	<b><i>Voice Message Notification</i></b> .....	<b>73</b>
249	<b>21.1</b>	<b>Symbol: <i>celf_mp_cs_voice_msg_notify</i></b> .....	<b>73</b>
250	21.1.1	Syntax .....	73
251	21.1.2	Argument .....	73
252	21.1.3	Return Value.....	73
253	21.1.4	Include File.....	73
254	21.1.5	Functional Description.....	73
255	<b>22.</b>	<b><i>Hold Tone Start</i></b> .....	<b>74</b>
256	<b>22.1</b>	<b>Symbol: <i>celf_mp_cs_hold_tone_start</i></b> .....	<b>74</b>
257	22.1.1	Syntax .....	74
258	22.1.2	Argument .....	74
259	22.1.3	Return Value.....	74
260	22.1.4	Include File.....	74
261	22.1.5	Functional Description.....	74
262	<b>23.</b>	<b><i>Hold Tone Stop</i></b> .....	<b>75</b>
263	<b>23.1</b>	<b>Symbol: <i>celf_mp_cs_hold_tone_stop</i></b> .....	<b>75</b>
264	23.1.1	Syntax .....	75
265	23.1.2	Argument .....	75
266	23.1.3	Return Value.....	75
267	23.1.4	Include File.....	75
268	23.1.5	Functional Description.....	75
269	<b>24.</b>	<b><i>Get 64K / AV Communication Status</i></b> .....	<b>76</b>

270	<b>24.1</b>	<b>Symbol: celf_mp_cs_get_UD_com_stat</b> .....	<b>76</b>
271	24.1.1	Syntax .....	76
272		Argument .....	76
273	24.1.2	.....	76
274	24.1.3	Return Value .....	76
275	24.1.4	Include File .....	76
276	24.1.5	Functional Description.....	76
277	<b>25.</b>	<b><i>Get internal/external AV Communication Status</i></b> .....	<b>77</b>
278	<b>25.1</b>	<b>Symbol: celf_mp_cs_get_AV_com_stat</b> .....	<b>77</b>
279	25.1.1	Syntax .....	77
280	25.1.2	Argument .....	77
281	25.1.3	Return Value .....	77
282	25.1.4	Include File .....	77
283	25.1.5	Functional Description.....	77
284	<b>26.</b>	<b><i>Get Communication Status</i></b> .....	<b>78</b>
285	<b>26.1</b>	<b>Symbol: celf_mp_cs_get_com_stat</b> .....	<b>78</b>
286	26.1.1	Syntax .....	78
287	26.1.2	Argument .....	78
288	26.1.3	Return Value .....	78
289	26.1.4	Include File .....	79
290	26.1.5	Functional Description.....	79
291	<b>27.</b>	<b><i>Start Line Status Notification</i></b> .....	<b>80</b>
292	<b>27.1</b>	<b>Symbol: celf_mp_cs_line_status_notification_start</b> .....	<b>80</b>
293	27.1.1	Syntax .....	80
294	27.1.2	Argument .....	80
295	27.1.3	Return Value .....	80
296	27.1.4	Include File .....	81
297	27.1.5	Functional Description.....	81
298	<b>28.</b>	<b><i>Stop Line Status Notification</i></b> .....	<b>82</b>
299	<b>28.1</b>	<b>Symbol: celf_mp_cs_line_status_notification_stop</b> .....	<b>82</b>
300	28.1.1	Syntax .....	82
301	28.1.2	Argument .....	82
302	28.1.3	Return Value .....	82
303	28.1.4	Include File .....	83
304	28.1.5	Functional Description.....	83
305	<b>29.</b>	<b><i>Get Reception Level</i></b> .....	<b>84</b>
306	<b>29.1</b>	<b>Symbol: celf_mp_cs_get_reception_level</b> .....	<b>84</b>
307	29.1.1	Syntax .....	84
308	29.1.2	Argument .....	84
309	29.1.3	Return Value .....	84
310	29.1.4	Include File .....	84
311	29.1.5	Functional Description.....	84
312	<b>30.</b>	<b><i>Get Line Status</i></b> .....	<b>85</b>
313	<b>30.1</b>	<b>Symbol: celf_mp_cs_get_line_status</b> .....	<b>85</b>
314	30.1.1	Syntax .....	85
315	30.1.2	Argument .....	85
316	30.1.3	Return Value .....	85



317	30.1.4	Include File .....	85
318	30.1.5	Functional Description.....	85
319	<b>31.</b>	<b><i>Get Coverage Status</i></b> .....	<b>86</b>
320	<b>31.1</b>	<b>Symbol: <i>celf_mp_cs_get_coverage_status</i></b> .....	<b>86</b>
321	31.1.1	Syntax .....	86
322	31.1.2	Argument .....	86
323	31.1.3	Return Value.....	86
324	31.1.4	Include File .....	86
325	31.1.5	Functional Description.....	86
326	<b>32.</b>	<b><i>Get Voice Mail Information</i></b> .....	<b>87</b>
327	<b>32.1</b>	<b>Symbol: <i>celf_mp_cs_get_vm_info</i></b> .....	<b>87</b>
328	32.1.1	Syntax .....	87
329	32.1.2	Argument .....	87
330	32.1.3	Return Value.....	87
331	32.1.4	Include File .....	87
332	32.1.5	Functional Description.....	87
333	<b>33.</b>	<b><i>Set Voice Mail Information</i></b> .....	<b>88</b>
334	<b>33.1</b>	<b>Symbol: <i>celf_mp_cs_set_vm_info</i></b> .....	<b>88</b>
335	33.1.1	Syntax .....	88
336	33.1.2	Argument .....	88
337	33.1.3	Return Value.....	88
338	33.1.4	Include File .....	88
339	33.1.5	Functional Description.....	88
340	<b>34.</b>	<b><i>Get Call Selection</i></b> .....	<b>89</b>
341	<b>34.1</b>	<b>Symbol: <i>celf_mp_cs_get_call_select</i></b> .....	<b>89</b>
342	34.1.1	Syntax.....	89
343	34.1.2	Argument .....	89
344	34.1.3	Return Value.....	89
345	34.1.4	Include File .....	89
346	34.1.5	Functional Description.....	89
347	<b>35.</b>	<b><i>Set Call Selection</i></b> .....	<b>90</b>
348	<b>35.1</b>	<b>Symbol: <i>celf_mp_cs_set_call_select</i></b> .....	<b>90</b>
349	35.1.1	Syntax .....	90
350	35.1.2	Argument .....	90
351	35.1.3	Return Value.....	90
352	35.1.4	Include File .....	90
353	35.1.5	Functional Description.....	90
354	<b>36.</b>	<b><i>Set Service Information</i></b> .....	<b>91</b>
355	<b>36.1</b>	<b>Symbol: <i>celf_mp_cs_set_service_info</i></b> .....	<b>91</b>
356	36.1.1	Syntax .....	91
357	36.1.2	Argument .....	91
358	36.1.3	Return Value.....	91
359	36.1.4	Include File .....	91
360	36.1.5	Functional Description.....	91
361	<b>37.</b>	<b><i>Get Service Information</i></b> .....	<b>93</b>
362	<b>37.1</b>	<b>Symbol: <i>celf_mp_cs_get_service_info</i></b> .....	<b>93</b>

363	37.1.1	Syntax .....	93
364	37.1.2	Argument .....	93
365	37.1.3	Return Value .....	93
366	37.1.4	Include File .....	93
367	37.1.5	Functional Description .....	93
368	<b>38.</b>	<b><i>Delete Service Information</i></b> .....	<b>94</b>
369	<b>38.1</b>	<b>Symbol: <i>celf_mp_cs_del_service_info</i></b> .....	<b>94</b>
370	38.1.1	Syntax .....	94
371	38.1.2	Argument .....	94
372	38.1.3	Return Value .....	94
373	38.1.4	Include File .....	94
374	38.1.5	Functional Description .....	94
375	<b>39.</b>	<b><i>Remove Service Information</i></b> .....	<b>95</b>
376	<b>39.1</b>	<b>Symbol: <i>celf_mp_cs_remove_all_service_info</i></b> .....	<b>95</b>
377	39.1.1	Syntax .....	95
378	39.1.2	Argument .....	95
379	39.1.3	Return Value .....	95
380	39.1.4	Include File .....	95
381	39.1.5	Functional Description .....	95
382	<b>40.</b>	<b><i>Set Response Message Settings</i></b> .....	<b>96</b>
383	<b>40.1</b>	<b>Symbol: <i>celf_mp_cs_set_resp_msg</i></b> .....	<b>96</b>
384	40.1.1	Syntax .....	96
385	40.1.2	Argument .....	96
386	40.1.3	Return Value .....	96
387	40.1.4	Include File .....	96
388	40.1.5	Functional Description .....	96
389	<b>41.</b>	<b><i>Get Response Message Settings</i></b> .....	<b>98</b>
390	<b>41.1</b>	<b>Symbol: <i>celf_mp_cs_get_resp_msg</i></b> .....	<b>98</b>
391	41.1.1	Syntax .....	98
392	41.1.2	Argument .....	98
393	41.1.3	Return Value .....	98
394	41.1.4	Include File .....	98
395	41.1.5	Functional Description .....	98
396	<b>42.</b>	<b><i>Delete Response Message Settings</i></b> .....	<b>99</b>
397	<b>42.1</b>	<b>Symbol: <i>celf_mp_cs_del_resp_msg</i></b> .....	<b>99</b>
398	42.1.1	Syntax .....	99
399	42.1.2	Argument .....	99
400	42.1.3	Return Value .....	99
401	42.1.4	Include File .....	99
402	42.1.5	Functional Description .....	99
403	<b>43.</b>	<b><i>Remove All Response Message Settings</i></b> .....	<b>100</b>
404	<b>43.1</b>	<b>Symbol: <i>celf_mp_cs_remove_all_resp_msg</i></b> .....	<b>100</b>
405	43.1.1	Syntax .....	100
406	43.1.2	Argument .....	100
407	43.1.3	Return Value .....	100
408	43.1.4	Include File .....	100
409	43.1.5	Functional Description .....	100

410	<b>44. Set Reconnection Tone .....</b>	<b>101</b>
411	<b>44.1 Symbol: celf_mp_cs_set_reconnection_tone .....</b>	<b>101</b>
412	44.1.1 Syntax .....	101
413	44.1.2 Argument .....	101
414	44.1.3 Return Value .....	101
415	44.1.4 Include File .....	101
416	44.1.5 Functional Description .....	101
417	<b>45. Get Reconnection Tone.....</b>	<b>102</b>
418	<b>45.1 Symbol: celf_mp_cs_get_reconnection_tone.....</b>	<b>102</b>
419	45.1.1 Syntax .....	102
420	45.1.2 Argument .....	102
421	45.1.3 Return Value .....	102
422	45.1.4 Include File .....	102
423	45.1.5 Functional Description .....	102
424	<b>46. Get Noise Cancel.....</b>	<b>103</b>
425	<b>46.1 Symbol: celf_mp_cs_get_noise_cancel.....</b>	<b>103</b>
426	46.1.1 Syntax .....	103
427	46.1.2 Argument .....	103
428	46.1.3 Return Value .....	103
429	46.1.4 Include File .....	103
430	46.1.5 Functional Description .....	103
431	<b>47. Set Noise Cancel.....</b>	<b>104</b>
432	<b>47.1 Symbol: celf_mp_cs_set_noise_cancel .....</b>	<b>104</b>
433	47.1.1 Syntax .....	104
434	47.1.2 Argument .....	104
435	47.1.3 Return Value .....	104
436	47.1.4 Include File .....	104
437	47.1.5 Functional Description .....	104
438	<b>48. Get Quality Alarm.....</b>	<b>105</b>
439	<b>48.1 Symbol: celf_mp_cs_get_quality_alarm.....</b>	<b>105</b>
440	48.1.1 Syntax .....	105
441	48.1.2 Argument .....	105
442	48.1.3 Return Value .....	105
443	48.1.4 Include File .....	105
444	48.1.5 Functional Description .....	105
445	<b>49. Set Quality Alarm.....</b>	<b>106</b>
446	<b>49.1 Symbol: celf_mp_cs_set_quality_alarm .....</b>	<b>106</b>
447	49.1.1 Syntax .....	106
448	49.1.2 Argument .....	106
449	49.1.3 Return Value .....	106
450	49.1.4 Include File .....	106
451	49.1.5 Functional Description .....	106
452	<b>50. Get Noise Cancel Permit.....</b>	<b>107</b>
453	<b>50.1 Symbol: celf_mp_cs_get_noise_cancel_permit .....</b>	<b>107</b>
454	50.1.1 Syntax .....	107
455	50.1.2 Argument .....	107

456	50.1.3	Return Value.....	107
457	50.1.4	Include File.....	107
458	50.1.5	Functional Description.....	107
459	<b>51.</b>	<b><i>Set High Priority communication mode.....</i></b>	<b>108</b>
460	<b>51.1</b>	<b>Symbol: <i>celf_mp_cs_set_hi_prio_com</i>.....</b>	<b>108</b>
461	51.1.1	Syntax.....	108
462	51.1.2	Argument.....	108
463	51.1.3	Return Value.....	108
464	51.1.4	Include File.....	108
465	51.1.5	Functional Description.....	108
466	<b>52.</b>	<b><i>Get Phone Answering Sound Activation.....</i></b>	<b>109</b>
467	<b>52.1</b>	<b>Symbol: <i>celf_mp_cs_get_vm_sound_status</i>.....</b>	<b>109</b>
468	52.1.1	Syntax.....	109
469	52.1.2	Argument.....	109
470	52.1.3	Return Value.....	109
471	52.1.4	Include File.....	109
472	52.1.5	Functional Description.....	109
473	<b>53.</b>	<b><i>Set Phone Answering Sound Activation.....</i></b>	<b>110</b>
474	<b>53.1</b>	<b>Symbol: <i>celf_mp_cs_set_vm_sound_status</i>.....</b>	<b>110</b>
475	53.1.1	Syntax.....	110
476	53.1.2	Argument.....	110
477	53.1.3	Return Value.....	110
478	53.1.4	Include File.....	110
479	53.1.5	Functional Description.....	110
480	<b>54.</b>	<b><i>Get Automatic Receive Status.....</i></b>	<b>111</b>
481	<b>54.1</b>	<b>Symbol: <i>celf_mp_cs_get_auto_rcv_status</i>.....</b>	<b>111</b>
482	54.1.1	Syntax.....	111
483	54.1.2	Argument.....	111
484	54.1.3	Return Value.....	111
485	54.1.4	Include File.....	111
486	54.1.5	Functional Description.....	111
487	<b>55.</b>	<b><i>Set Automatic Receive Status.....</i></b>	<b>112</b>
488	<b>55.1</b>	<b>Symbol: <i>celf_mp_cs_set_auto_rcv_status</i>.....</b>	<b>112</b>
489	55.1.1	Syntax.....	112
490	55.1.2	Argument.....	112
491	55.1.3	Return Value.....	112
492	55.1.4	Include File.....	112
493	55.1.5	Functional Description.....	112
494	<b>56.</b>	<b><i>Get Automatic Timer.....</i></b>	<b>113</b>
495	<b>56.1</b>	<b>Symbol: <i>celf_mp_cs_get_auto_timer</i>.....</b>	<b>113</b>
496	56.1.1	Syntax.....	113
497	56.1.2	Argument.....	113
498	56.1.3	Return Value.....	113
499	56.1.4	Include File.....	113
500	56.1.5	Functional Description.....	113
501	<b>57.</b>	<b><i>Set Automatic Timer.....</i></b>	<b>114</b>

502	<b>57.1 Symbol: celf_mp_cs_set_auto_timer.....</b>	<b>114</b>
503	57.1.1 Syntax .....	114
504	57.1.2 Argument .....	114
505	57.1.3 Return Value.....	114
506	57.1.4 Include File.....	114
507	57.1.5 Functional Description.....	114
508	<b>58. Get Reset Date .....</b>	<b>115</b>
509	<b>58.1 Symbol: celf_mp_cs_get_reset_date .....</b>	<b>115</b>
510	58.1.1 Syntax .....	115
511	58.1.2 Argument .....	115
512	58.1.3 Return Value.....	115
513	58.1.4 Include File.....	115
514	58.1.5 Functional Description.....	115
515	<b>59. Set Reset Date.....</b>	<b>116</b>
516	<b>59.1 Symbol: celf_mp_cs_set_reset_date.....</b>	<b>116</b>
517	59.1.1 Syntax .....	116
518	59.1.2 Argument .....	116
519	59.1.3 Return Value.....	116
520	59.1.4 Include File.....	116
521	59.1.5 Functional Description.....	116
522	<b>60. Get Call Silent Time.....</b>	<b>117</b>
523	<b>60.1 Symbol: celf_mp_cs_get_call_silent_time .....</b>	<b>117</b>
524	60.1.1 Syntax .....	117
525	60.1.2 Argument .....	117
526	60.1.3 Return Value.....	117
527	60.1.4 Include File.....	117
528	60.1.5 Functional Description.....	117
529	<b>61. Set Call Silent Time.....</b>	<b>118</b>
530	<b>61.1 Symbol: celf_mp_cs_set_call_silent_time.....</b>	<b>118</b>
531	61.1.1 Syntax .....	118
532	61.1.2 Argument.....	118
533	61.1.3 Return Value.....	118
534	61.1.4 Include File.....	118
535	61.1.5 Functional Description.....	118
536	<b>62. Get Call Recorded.....</b>	<b>119</b>
537	<b>62.1 Symbol: celf_mp_cs_get_call_recorded.....</b>	<b>119</b>
538	62.1.1 Syntax .....	119
539	62.1.2 Argument .....	119
540	62.1.3 Return Value.....	119
541	62.1.4 Include File.....	119
542	62.1.5 Functional Description.....	119
543	<b>63. Set Call Recorded.....</b>	<b>120</b>
544	<b>63.1 Symbol: celf_mp_cs_set_call_recorded .....</b>	<b>120</b>
545	63.1.1 Syntax .....	120
546	63.1.2 Argument .....	120
547	63.1.3 Return Value.....	120
548	63.1.4 Include File.....	120

549      63.1.5 Functional Description..... 120  
550

DRAFT

551 **0. Introduction**

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

554 Circuit Switched Communication Service includes

- 555     • Voice communication service,  
556     • Video communication service, and  
557     • Digital data Communication service.

DRAFT

558

559 **References**

560 **0.1.1 Normative**

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

563 RFC 2234: “Augmented BNF for Syntax Specifications: ABNF”. D. Crocker, Ed., P. Overell. November  
564 1997, [URL:http://www.ietf.org/rfc/rfc2234.txt](http://www.ietf.org/rfc/rfc2234.txt)

565

566 **0.1.2 Informative**

DRAFT



567 **1. Primitives**

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

569 **1.1 Constants**

570 **1.1.1 Line type**

571 CELF\_MP\_CS\_LINE\_WCDMA WCDMA

572 CELF\_MP\_CS\_LINE\_UMTS UMTS

573

574 **1.1.2 Dial Number**

575 Dial number of the other party

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

577 CELF\_MP\_CS\_DIAL\_MAX is 45.

578 **1.1.3 TAF address**

579 TAF address is the connection ID in TAF (Terminal Adaptation Function).

580 This is external to the CELF MPP specification.

581 See the following documents for further details: 3GPP TS 27.001 / 3GPP TS 24.002

582 **1.1.4 Supplementary service**

583 Service Info Name

584 CELF\_SRVINFO\_TITLE 21

585 Dial data for accessing the service

586 CELF\_SRVINFO\_DATA 40

587 Response Message Name

588 CELF\_RESMSG\_TITLE 21

589 Response data for accessing the service

590 CELF\_RESMSG\_DATA 40

591

## 592 1.2 Enums

### 593 1.2.1 Voice communication status (CelfMpCsComStatus)

594 In this operation mode, following multiple calls can be handled by a mobile phone, simultaneously. Each  
595 state of handling calls is:

- 596 • Holding a receiving call
- 597 • Talk over the phone
- 598 • Receive another incoming call

599 The number of handling states is three.

600 This is called the multiple calls.

601

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

#### 603 1.2.1.1 Condition: only one call

```
604 enum CelfMpCsComStatus {
605     CELF_MP_CS_COM_STATUS_WAIT,           // Standby
606     CELF_MP_CS_COM_STATUS_RCV,           // Under incoming
607     CELF_MP_CS_COM_STATUS_TRN,           // Under outgoing
608     CELF_MP_CS_COM_STATUS_DLV,           // Under calling
609     CELF_MP_CS_COM_STATUS_TLK,           // Under conversation
610     CELF_MP_CS_COM_STATUS_HLD,           // Under response hold
611     /* This status is (a) that incoming call was received, and (b) that this incoming call cannot transit to
612     conversation status because of the mobile phone. */
613     CELF_MP_CS_COM_STATUS_RLS           // Under release
614 }

```

#### 615 1.2.1.2 Condition: two call

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

```
617 CELF_MP_CS_COM_STATUS_TLK_RCV:           // Under conversation and incoming
618 CELF_MP_CS_COM_STATUS_TLK_TRN:           // Under conversation and outgoing
619 CELF_MP_CS_COM_STATUS_TLK_DLV:           // Under conversation and calling
620 CELF_MP_CS_COM_STATUS_TLK_RSV:           // Under conversation and hold
621 CELF_MP_CS_COM_STATUS_TLK_RLS:           // Under conversation and release

```

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

623 incoming.

```
624 CELF_MP_CS_COM_STATUS_TLK_RSV_RCV: // Under conversation, hold, and incoming

```

#### 625 1.2.1.3 Condition: only one AV call

```
626 CELF_MP_CS_COM_STATUS_RCV_AV:           // Under incoming of an AV call
627 CELF_MP_CS_COM_STATUS_TRN_AV:           // Under outgoing of an AV call

```

Classification: *Circuit Switched Service*

628 CELF\_MP\_CS\_COM\_STATUS\_DLV\_AV: // Under calling of an AV call  
629 CELF\_MP\_CS\_COM\_STATUS\_TLK\_AV: // Under conversation of an AV call  
630 CELF\_MP\_CS\_COM\_STATUS\_HLD\_AV: // Under response hold of an AV call  
631 CELF\_MP\_CS\_COM\_STATUS\_RLS\_AV: // Under release of an AV call

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

- 633 (a) that one call is in incoming and another call is in outgoing,
- 634 (b) that two call are both in conversation,
- 635 (c) that two call are in hold and other call is in conversation, and so on.

636

### 637 1.2.2 Forwarding result (CelfMpCSFwResult)

638 CELF\_MP\_CS\_OK // Successful forwarding  
639 CELF\_MP\_CS\_ERR // Forwarding failure

640

### 641 1.2.3 Forwarding result details (CelfMpCsFwError)

642 Set only at forwarding failure.

643 CELF\_MP\_CS\_FW\_ERROR\_NO\_JOIN // Service is not contracted.  
644 CELF\_MP\_CS\_FW\_ERROR\_NO\_SETDATA // the forwarded destination is not registered.  
645 CELF\_MP\_CS\_FW\_ERROR\_ETC // Others

646

### 647 1.2.4 Communication type (CelfMpCsBtype)

648 CELF\_MP\_CS\_BTYPE\_CS\_NONE // None (unfixed)  
649 CELF\_MP\_CS\_BTYPE\_CS\_ANY // Not Specified  
650 CELF\_MP\_CS\_BTYPE\_CS\_VOICE // Voice  
651 CELF\_MP\_CS\_BTYPE\_CS\_UD32UD // 32K communication  
652 CELF\_MP\_CS\_BTYPE\_CS\_UD64UD // 64K communication  
653 CELF\_MP\_CS\_BTYPE\_CS\_AV32AV // 32K communication  
654 CELF\_MP\_CS\_BTYPE\_CS\_AV64AV // 64K communication

655

### 656 1.2.5 Call Reference Status

657 CELF\_MP\_CS\_USED: // "CN\_No" – Connection Number - is valid.  
658 CELF\_MP\_CS\_UNUSED: // "CN\_No" – Connection Number - is not valid.

659 In some cases the Call reference status is unused, indicated by CELF\_MP\_CS\_UNUSED. If so, there is no  
660 connection between this mobile phone and other party and all data is void.

661

### 662 1.2.6 Call Status

663 Call status for this mobile phone

**Classification: *Circuit Switched Service***

664 CELF\_MP\_CS\_CHAN\_NULL: // Vacant  
665 CELF\_MP\_CS\_CHAN\_OFF: // Off-hook  
666 CELF\_MP\_CS\_CHAN\_TRN: // Outgoing call  
667 CELF\_MP\_CS\_CHAN\_DLV: // Calling  
668 CELF\_MP\_CS\_CHAN\_CV: // Incoming call  
669 CELF\_MP\_CS\_CHAN\_REQ\_T: // Response (conversation)  
670 (The status of responding mobile phone is conversation.)  
671 CELF\_MP\_CS\_CHAN\_ACT: // Under conversation  
672 CELF\_MP\_CS\_CHAN\_REQ\_H: // Response (hold)  
673 (The status of responding mobile phone is hold.)  
674 CELF\_MP\_CS\_CHAN\_HLD: // Hold response  
675 CELF\_MP\_CS\_CHAN\_RSV: // Under hold  
676 CELF\_MP\_CS\_CHAN\_REL: // Under release

677

### 678 1.2.7 Existence of continuation data

679 CELF\_MP\_CS\_ON: // valid below data  
680 CELF\_MP\_CS\_OFF: // non valid below data

681 The below data, from "Calling\_Dial" to "cause", are valid data if the call status is incoming or conversation  
682 and incoming call.

683

684

### 685 1.2.8 Busy Tone sound flag

686 Whether Busy Tone (engaged tone) sounds in this phone, or not

687 CELF\_MP\_CS\_SOUND\_BT\_ON: // BT tone sounds.  
688 CELF\_MP\_CS\_SOUND\_BT\_OFF: // BT tone is being stopped.

689

690

### 691 1.2.9 Cause of NoCLI

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

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

694 CELF\_MP\_CS\_NOCL\_NOSRV: // service is not supported.  
695 CELF\_MP\_CS\_NOCL\_USER: // user rejects to display.  
696 CELF\_MP\_CS\_NOCL\_INTRACTSRV: // service conflicts.  
697 CELF\_MP\_CS\_NOCL\_PAYPHONE: // origination is from a public phone.

698 This data is valid, when next data "num\_presentation\_indicator", is that Display is impossible.

699

## 700 1.2.10 Dial number / Redirect number display indicator

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

702 CELF\_MP\_CS\_PRSENT\_IND\_ALLOWED: // Displayable  
703 CELF\_MP\_CS\_PRSENT\_IND\_RESTRICTED: // Impossible to display  
704 CELF\_MP\_CS\_PRSENT\_IND\_NOT\_AVAILABLE: // Displayable number does not exist.  
705 CELF\_MP\_CS\_PRSENT\_IND\_RESERVE: // Reservation

706

## 707 1.2.11 Signal information (CelfMpCsSignal)

708 The type of tone of this phone

709 CELF\_MP\_CS\_SIGNAL\_DIAL\_TONE\_ON: // Dial tone on  
710 CELF\_MP\_CS\_SIGNAL\_RINGBACK\_TONE\_ON: // Ring back tone on  
711 CELF\_MP\_CS\_SIGNAL\_INTERCEPT\_TONE\_ON: // Intercept tone on  
712 CELF\_MP\_CS\_SIGNAL\_NW\_CONGESTION\_TONE\_ON: // Network congestion tone on  
713 CELF\_MP\_CS\_SIGNAL\_BUSY\_TONE\_ON: // Busy tone on  
714 CELF\_MP\_CS\_SIGNAL\_CONFIRM\_TONE\_ON: // Confirm tone on  
715 CELF\_MP\_CS\_SIGNAL\_ANSWER\_TONE\_ON: // Answer tone on  
716 CELF\_MP\_CS\_SIGNAL\_CALLWAITING\_TONE\_ON: // Call waiting tone on  
717 CELF\_MP\_CS\_SIGNAL\_OFFHK\_WARNING\_TONE\_ON: // Off-hook warning tone on  
718 CELF\_MP\_CS\_SIGNAL\_TONES\_OFF: // Tones off  
719 CELF\_MP\_CS\_SIGNAL\_ALERTING\_OFF: // Alerting off  
720 CELF\_MP\_CS\_SIGNAL\_UNSETTING: // Signal information is not set.

## 721 1.2.12 Originating Number notification (CelfMpCsNotice)

722 Whether the originating dial number is notified or not.

723 CELF\_MP\_CS\_NOTICE\_ON: // Notified  
724 CELF\_MP\_CS\_NOTICE\_OFF: // Not notified  
725 CELF\_MP\_CS\_NOTICE\_NOSET: // No setting

## 726 1.2.13 Line status (CelfMpCsLineStatus)

727 CELF\_MP\_CS\_LINE\_STATUS\_OUT: // Out-of-communication area  
728 CELF\_MP\_CS\_LINE\_STATUS\_IN: // Within-communication area

## 729 1.2.14 Normal and emergency originating restriction

730 CELF\_MP\_CS\_LINE\_RESTRICT\_DATA\_ON // With originating restriction  
731 CELF\_MP\_CS\_LINE\_RESTRICT\_DATA\_OFF // Without originating restriction

## 732 1.2.15 Receive level (CelfMpCsRSSIlevel)

733 CELF\_MP\_CS\_RSSI\_LEVEL\_0: // Receive level 0  
734 CELF\_MP\_CS\_RSSI\_LEVEL\_1: // Receive level 1

735 CELF\_MP\_CS\_RSSI\_LEVEL\_2: // Receive level 2

736 CELF\_MP\_CS\_RSSI\_LEVEL\_3: // Receive level 3

### 737 1.2.16 Area status information (CelfMpCsLineCvrStatus)

738 CELF\_MP\_CS\_LINE\_CVR\_STATUS\_IN IN

739 CELF\_MP\_CS\_LINE\_CVR\_STATUS\_OUT OUT

740

### 741 1.2.17 RRC mode (CelfMpCsLineRRCMODE)

742 CELF\_MP\_CS\_LINE\_RRC\_MODE\_IDLE // idle-mode

743 CELF\_MP\_CS\_LINE\_RRC\_MODE\_UTRAN // utran-connected-mode

744

745 Network identification information

746 CELF\_MP\_CS\_LINE\_NETWORK\_HOME // home

747 CELF\_MP\_CS\_LINE\_NETWORK\_VISIT // visit

748 CELF\_MP\_CS\_LINE\_NO\_DATA // No data

749

### 750 1.2.18 Service status (CelfMpLineSrvStatus)

751 CELF\_MP\_LINE\_SRV\_STATUS\_CS // CS is in service.

752 CELF\_MP\_LINE\_SRV\_STATUS\_PS // PS is in service.

753 CELF\_MP\_LINE\_SRV\_STATUS\_CSPS // CS and PS are in service.

754 CELF\_MP\_LINE\_NO\_DATA // No data

755 CS is the circuit switched communication service, and

756 PS is the packet switched communication service.

757

### 758 1.2.19 Restriction status (CelfMpCsLineRestrict)

759 CELF\_MP\_CS\_LINE\_RESTRICT\_ON // In traffic restriction

760 CELF\_MP\_CS\_LINE\_RESTRICT\_OFF // Out of traffic restriction

761

### 762 1.2.20 Identifying flag (CelfMpCsFlag)

763 enum CelfMpCsFlag {

764 CELF\_MP\_CS\_NO\_FLAG, // no Flag

765 CELF\_MP\_CS\_OPT\_FLAG, // special number

766 CELF\_MP\_CS\_USSD\_FLAG // USSD number

767 }

### 768 1.2.21 Notification Set (CelfMpCsNotifySet)

769 CELF\_MP\_CS\_CLASS\_COM\_STATUS // Voice communication status notification

```
770 CELF_MP_CS_CLASS_TLK_TIME           // Call duration notification
771 CELF_MP_CS_CLASS_DISC_CAUSE        // Disconnection cause notification
772 CELF_MP_CS_CLASS_FW_RESULT        // Call forwarding result notification
773 CELF_MP_CS_CLASS_OFFHK_TO         // Off-hook originating timeout notification
774
```

### 775 1.2.22 Event Structure Category

```
776 enum {
777     VoiceNotify
778 }
```

### 779 1.2.23 Event Structure Subtype

```
780 enum {
781     VoiceNotify_ConnInfo
782     VoiceNotify_TelCallTime
783     VoiceNotify_DiscCause
784     VoiceNotify_FW_Result
785     VoiceNotify_OffHk_Trn
786     DCF_Event_type
787     VoiceNotify_AreaInfo
788     VoiceNotify_RssiLevel
789 }
```

### 790 1.2.24 Call Number (CelfMpCallNo)

```
791 int CelfMpCallNo // reference to call number 0..255
792
```

### 793 1.2.25 DCF Event Set (CelfMpCsDCFSet)

```
794 CELF_MP_CS_DCF_DISP // Display-related message
795 CELF_MP_CS_DCF_HISTORY // History-related message
796 CELF_MP_CS_DCF_TONE1 // Tone 1-related message
797 CELF_MP_CS_DCF_TONE2 // Tone 2-related message
798 CELF_MP_CS_DCF_ETC // Other messages
799 CELF_MP_CS_CLASS_ALL // All notified
800
```

### 801 1.2.26 Voice message (CelfMpCsRecMsg)

```
802 CELF_MP_CS_REC_MSG_START // Start of a voice message
803 CELF_MP_CS_REC_MSG_STOP // Stop of a voice message
804
```

805 **1.2.27 Off Hook Option (CelfMpCsOffHk)**

806 CELF\_MP\_CS\_OFFHK\_AUTO // Automatic transmission  
807 CELF\_MP\_CS\_OFFHK\_MANUAL // Manual transmission

808

809 **1.2.28 64K/AV Communication (CelfMpCsUDComStatus)**

810 CELF\_MP\_CS\_UD\_STOP // Under stop  
811 CELF\_MP\_CS\_UD\_RUN // Under communication  
812 CELF\_MP\_CS\_UD\_CALLED // Under incoming  
813 CELF\_MP\_CS\_UD\_CALLING // Under outgoing  
814 CELF\_MP\_CS\_UD\_DISCONNECT // Under disconnection  
815 CELF\_MP\_CS\_UD\_CALLING\_ALERT // Under calling  
816 CELF\_MP\_CS\_UD\_HOLD // Under hold  
817 CELF\_MP\_CS\_UD\_ERR // Error in UD communication

818

819 **1.2.29 AV Communication (CelfMpAVComStatus)**

820 CELF\_MP\_CS\_AV\_IN\_STOP // Under stop  
821 CELF\_MP\_CS\_AV\_IN\_RUN // Under communication  
822 CELF\_MP\_CS\_AV\_IN\_CALLED // Under incoming  
823 CELF\_MP\_CS\_AV\_IN\_CALLING // Under outgoing  
824 CELF\_MP\_CS\_AV\_IN\_DISCONNECT // Under disconnection  
825 CELF\_MP\_CS\_AV\_IN\_CALLING\_ALERT // Under calling  
826 CELF\_MP\_CS\_UD\_IN\_HOLD // Under hold  
827 CELF\_MP\_CS\_AV\_OUT\_STOP // Under stop  
828 CELF\_MP\_CS\_AV\_OUT\_RUN // Under communication  
829 CELF\_MP\_CS\_AV\_OUT\_CALLED // Under incoming  
830 CELF\_MP\_CS\_AV\_OUT\_CALLING // Under outgoing  
831 CELF\_MP\_CS\_AV\_OUT\_DISCONNECT // Under disconnection  
832 CELF\_MP\_CS\_AV\_OUT\_CALLING\_ALERT // Under calling  
833 CELF\_MP\_CS\_UD\_OUT\_HOLD // Under hold  
834 CELF\_MP\_CS\_UD\_ERR // Error in UD communication

835

836 **1.2.30 Receive Scene Events (CelfMpCsRcvScene)**

837 CELF\_MP\_CS\_RCV\_SCENE\_COMPETE\_TRN // Outgoing conflict  
838 CELF\_MP\_CS\_RCV\_SCENE\_RSV\_RETURN // Incoming hold call  
839 CELF\_MP\_CS\_RCV\_SCENE\_CALL\_BACK // Re-incoming



840 CELF\_MP\_CS\_RCV\_SCENE\_NORMAL // Normal

841 CELF\_MP\_CS\_RCV\_SCENE\_NON // Unset

842

### 843 1.2.31 Line Monitoring (CelfMpCsMtype)

844 CELF\_MP\_CS\_MONITOR\_LINE\_STATUS // Line status change notification

845 CELF\_MP\_CS\_MONITOR\_RESTRICT // Restriction status change notification

846 CELF\_MP\_CS\_MONITOR\_RSSI // Receive level change notification

847 CELF\_MP\_CS\_MONITOR\_ALL // All notified

### 848 1.2.32 Reception Level (CelfMpReceptionLevel)

849 CELF\_MP\_CS\_RSSI\_LEVEL\_0 // Receive level 0

850 CELF\_MP\_CS\_RSSI\_LEVEL\_1 // Receive level 1

851 CELF\_MP\_CS\_RSSI\_LEVEL\_2 // Receive level 2

852 CELF\_MP\_CS\_RSSI\_LEVEL\_3 // Receive level 3

853

### 854 1.2.33 Coverage Indicators (CelfMpCsCoverage)

855 CELF\_MP\_CS\_LINE\_STATUS\_IN // Within-communication area

856 CELF\_MP\_CS\_LINE\_STATUS\_OUT // Out-of-communication area

857

### 858 1.2.34 Incoming Call Selection (CelfMpCallSelect)

859 CELF\_MP\_CS\_INCOMING\_VOICE\_ANSWERING // Forward to the phone-answering message

860 CELF\_MP\_CS\_INCOMING\_FORWARD // Forward

861 CELF\_MP\_CS\_INCOMING\_REJECT // Reject (disconnect)

862 CELF\_MP\_CS\_INCOMING\_NORMAL // Receipt of an incoming call (normal incoming)

863

### 864 1.2.35 Registration number (CelfMpRegNum)

865 int CelfMpRegNum // Registration number: 1 to 10

866

### 867 1.2.36 Service Data (CelfMpCsSrvData)

868 char\* CelfMpCsSrvData // Pointer to supplementary service data

869

### 870 1.2.37 Reconnection Tone (CelfMpCsReconnectionTone)

871 CELF\_MP\_CS\_RECONN\_ON\_T\_OFF // Tone OFF

872 CELF\_MP\_CS\_RECONN\_ON\_T\_LOW // Tone ON low tone

873 CELF\_MP\_CS\_RECONN\_ON\_T\_HI // Tone ON high tone

874

875 **1.2.38 Noise Canceling (CelfMpCsNoiseCancel)**

876 CELF\_MP\_CS\_ON: Noise canceller ON

877 CELF\_MP\_CS\_OFF: Noise canceller OFF

878

879 **1.2.39 Quality Alarm (CelfMpCsQualAlarm)**

880 CELF\_MP\_CS\_QUALITY\_ALM\_OFF // Quality alarm OFF

881 CELF\_MP\_CS\_QUALITY\_ALM\_LOW // Quality alarm ON low tone

882 CELF\_MP\_CS\_QUALITY\_ALM\_HI // Quality alarm ON high tone

883

884 **1.2.40 Reconnection Tone Priority (CelfMpCsHiPrioCom)**

885 CELF\_MP\_CS\_COMPRI\_NONE // No setting

886 CELF\_MP\_CS\_COMPRI\_VOICE // Voice

887 CELF\_MP\_CS\_COMPRI\_PACKET // Packet

888

889 **1.2.41 Message Sound settings (CelfMpCsVmSound)**

890 CELF\_MP\_CS\_ON // Message sound ON

891 CELF\_MP\_CS\_OFF // Message sound OFF

892

893 **1.2.42 Incoming Call Auto Receive (CelfMpCsAutoRcv)**

894 CELF\_MP\_CS\_ON // Automatic incoming call ON

895 CELF\_MP\_CS\_OFF // Automatic incoming call OFF

## 896 1.3 Data Types and Structures

### 897 1.3.1 Circuit switched status notification event structure

898 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

899 category = VoiceNotify;

900 subtype = VoiceNotify\_ConnInfo;

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

902 The field “data” carries:

```
903     CELF_MP_CS_RES_CHG_INF   res_chg_inf;    // to be used in the case of:  
904                                     // Restriction display information structure
```

### 906 1.3.2 Call duration notification event structure

907 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

908 category = VoiceNotify;

909 subtype = VoiceNotify\_TelCallTime;

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

911 The field “data” is unused.

912

### 913 1.3.3 Disconnection cause notification event structure

914 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

915 category = VoiceNotify;

916 subtype = VoiceNotify\_DiscCause;

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

918 The field “data” carries:

```
919     CelfMpCsDiscCause cme; //Disconnection cause information structure
```

920

### 921 1.3.4 Disconnection cause information structure

```
922 typedef struct {
```

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

```
924     unsigned char code; //Result code
```

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

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

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

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

```
929 } CelfMpCsDiscCause;
```

930

### 931 1.3.5 Forwarding result notification event structure

932 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

933 category = VoiceNotify;

934 subtype = VoiceNotify\_FW\_Result

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

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

937 The field “data” carries:

```
938         CelfMpCsFwResult      fw_result; // Forwarding result structure
```

939

### 940 1.3.6 Forwarding result structure (CelfMpCsFwResult)

```
941 typedef struct {
```

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

```
943 } CelfMpCsFwResult;
```

944

### 945 1.3.7 Off-hook transmission timeout event structure

946 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

947 category = VoiceNotify;

948 subtype = VoiceNotify\_OffHk\_Trn

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

950 The field “data” is unused.

951

### 952 1.3.8 Connection Destination Information (CelfMpConnectInfo)

```
953 typedef struct {
```

```
954     int      CN_No;           // Call reference
```

```
955     int      CN_status;
```

```
956     int      continue_flag;
```

```
957     unsigned char Calling_Dial [CELF_MP_CS_DIAL_MAX+1];
```

```
958     unsigned char Called_Dial [CELF_MP_CS_DIAL_MAX+1];
```

```
959     unsigned char BTsound_inf;
```

```
960     CelfMpCsBtype bc_type;
```

```
961     unsigned char[10] taf_address;
```

```
962     unsigned char cause_of_NoCLI;
```

```
963     unsigned char num_presentation_indicator;
```

```
964     unsigned char redirectnum [CELF_MP_CS_DIAL_MAX+1];
```

```
965     unsigned char redirect_presentation_indicator;
```

```
966 unsigned char      signal;  
967 CelfMpCsDiscCause  cause; // Disconnection cause information structure  
968 } CelfMpCsConnectInf  
969
```

### 1.3.9 Connection Request (CelfMpCsConReq)

```
971 typedef struct {  
972     CelfMpCsBtype      type;  
973     unsigned char *    dial_buf;  
974     int                 dial_len;  
975     CelfMpCsNotice     notice;  
976     unsigned char *    subaddr_buf;  
977     int                 subaddr_len;  
978 } CelfConReq  
979
```

### 1.3.10 Redirection number

```
981 Destination number of call transfer.  
982 redirectnum [CELF_MP_CS_DIAL_MAX+1]  
983
```

### 1.3.11 Channel Number Information (CelfMpCsChanNum)

```
985 CelfMpCsChanNum is used to hold call reference information.  
986 If a channel is not used, CELF_MP_CS_CHAN_NOUSE is set as the call reference.  
987
```

```
988 typedef struct {  
989     int ChanNum_00      // Call reference information 00  
990     int ChanNum_01      // Call reference information 01  
991     int ChanNum_02      // Call reference information 02  
992 } CelfMpCsChanNum  
993
```

### 1.3.12 Channel not in use Flag

```
995 int CELF_MP_CS_CHAN_NOUSE      // usually holds the call reference if channels are not used.  
996
```

### 1.3.13 DCF Event Structure

```
998 In this sub-section, the associated data structure is CELFMPEVENT with the following values:  
999 category = VoiceNotify;  
1000 subtype = DCF_Event_type;
```

1001 The value of field “info” is the notification type.  
1002 The value of field “subinfo” is the bearer type  
1003 The field “data” carries:  
1004         DCF message structure corresponding to report types.  
1005

### 1006 1.3.14 Line status change notification event structure

1007 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

1008 category = VoiceNotify;  
1009 subtype = VoiceNotify\_AreaInfo;  
1010 The value of field “info” is the line status.  
1011 The value of field “subinfo” is the line type.  
1012 The field “data” is unused.

### 1013 1.3.15 Restriction display information structure 1014 (CelfMpCsResChgInf)

```
1015 typedef struct {  
1016     unsigned char NcRestriction; //Normal originating restriction  
1017     unsigned char ServiceStatus; //Service status  
1018     unsigned char EcRestriction; //Emergency originating restriction  
1019 } CelfMpCsResChgInf;
```

### 1020 1.3.16 Receive level change notification event structure

1021 In this sub-section, the associated data structure is CELFMPEVENT with the following values:

1022 category = VoiceNotify;  
1023 subtype = VoiceNotify\_RssiLevel;  
1024 The value of field “info” is the receive level.  
1025 The value of field “subinfo” is the line type.  
1026 The field “data” is unused.

### 1027 1.3.17 Line Status structure (CelfMpCsAreaRefChgInf)

```
1028 typedef struct {  
1029     CelfMpCsLineStatus      LineStatus ;           //Line status  
1030     CelfMpCsLineStatus      CoverageStatus ;       //Service status  
1031     CelfMpCsLineRRCMode RRC mode ;                 //RRC mode  
1032     unsigned char           Network ;               //Network identification information  
1033     unsigned char           unused;                 //unused  
1034     CelfMpCsLineCvrStatus   ServiceStatus_AREA ;  //Area status information  
1035     CelfMpCsLineRestrict    RestrictStatus ;      //Restriction status  
1036     unsigned char           NcRestriction ;        //Normal originating restriction
```

```
1037 unsigned char      ServiceStatus_RES ;      //Service status
1038 unsigned char      EcRestriction ;          //Emergency originating restriction
1039 } CelfMpCsAreaRefChgInf ;
1040
```

### 1.3.18 Supplementary service data structure (CelfMpCsAddsrvData)

```
1043 typedef struct {
1044     CelfMpCsFlag  flag ;
1045     char title[CELF_SRVINFO_TITLE];          // Supplementary service name CELF_SRVINFO_TITLE=21
1046     char send_no[CELF_SRVINFO_DATA];        // Dial data for accessing the service
1047                                             // CELF_SRVINFO_DATA=40
1048 } CelfMpCsAddsrvData;
1049
```

### 1.3.19 Response Message Data Structure (CelfMpCsResponseMsgData)

1052 The supplementary response message information is the service name and Dial data, which is response  
1053 message to send the network.

```
1054
1055 typedef struct {
1056     unsigned char  title[CELF_RESMSG_TITLE] ;          // Service name
1057     unsigned char  res_msg[CELF_RESMSG_DATA];          // Dial data
1058 } CelfMpCsResponseMsgData;
1059
```

### 1.3.20 Line Status Extension (CelfMpCsLineStatusEx)

```
1060 unsigned char* CelfMpCsLineStatusEx;          // data for additional line status information
1061
```

### 1.3.21 Number of stored messages (CelfMpCsVMNum)

```
1062 int CelfMpCsVMNum
1063
1064
```

### 1.3.22 Date Format Structure (CelfMpCsDate)

```
1065
1066 typedef struct {
1067     unsigned char  Month
1068     unsigned char  Day
1069     unsigned char  Hour
1070     unsigned char  Minute
1071 } CelfMpCsDate
1072
```

1073 **1.3.23 Dial Buffer (CelfMpCsDialBuffer)**

1074 char\* CelfMpCsDialBuffer

1075 **1.3.24 Dial Buffer Length (CelfMpCsDialLen)**

1076 int CelfMpCsDialLen

1077 **1.3.25 Multi Party Operation (CelfMpCsMop)**

1078 CELF\_MP\_CS\_MOP\_RSV\_DISC: // Disconnect the hold call

1079 CELF\_MP\_CS\_MOP\_DISC\_AND\_RSP: // Response after disconnection

1080 CELF\_MP\_CS\_MOP\_RSV\_AND\_RSP: // Response after hold (including operation for switching a  
1081 call)

1082 CELF\_MP\_CS\_MOP\_CR\_DISC: // Disconnect call specified by the call reference

1083 **1.3.26 Timer Value (CelfMpCsTimer)**

1084 int CelfMpCsTimer // value 1 .. 120 seconds

1085

DRAFT



1086 **1.4 Events Type**1087 **1.4.1 DCF Event Type**

1088	VoiceNotify_DCF_Dis	Display-related message
1089	VoiceNotify_DCF_History	History-related message
1090	VoiceNotify_DCF_Tone1	Tone 1-related message
1091	VoiceNotify_DCF_Tone2	Tone 2-related message
1092	VoiceNotify_DCF_ETC	Other messages

1093

1094 **1.4.2 CCP Notification type**

1095	CELF_MP_CS_CCP_CALLING_START_REQ	Notification of starting display during CCP outgoing
1096	CELF_MP_CS_CCP_CALLED_START_IND	Notification of starting display during CCP incoming
1097	CELF_MP_CS_CCP_CALLING_ALERTING_IND	Notification of starting display during CCP calling
1098	CELF_MP_CS_CCP_CONNECT_START_RSP	Notification of starting display during CCP connection
1099		
1100	CELF_MP_CS_CCP_CONNECT_START_IND	Notification of starting display during CCP communication
1101		
1102	CELF_MP_CS_CCP_RELEASE_IND	Notification of ending CCP display
1103	CELF_MP_CS_CCP_DISCONNECT_REQ	Notification of starting CCP disconnection (on a mobile device) display
1104		
1105	CELF_MP_CS_CCP_DISCONNECT_START_IND	Notification of starting CCP disconnection (on a network) display
1106		
1107	CELF_MP_CS_CCP_CALLING_REJ_IND	Notification of rejecting CCP outgoing
1108	CELF_MP_CS_CCP_HOLD_CNF	Notification of CCP hold
1109	CELF_MP_CS_CCP_RETREIVE_CNF	Notification of releasing CCP hold
1110	CELF_MP_CS_CCP_CALLING_SETUP_REQ	Notification of registering CCP outgoing call history
1111	CELF_MP_CS_CCP_CALLED_REJ_REQ	Notification of registering CCP absence incoming call history
1112		
1113	CELF_MP_CS_CCP_CALLED_SETUP_RSP	Notification of registering CCP incoming call history
1114	CELF_MP_CS_CCP_RGT_START	Notification of CCP RGT start
1115	CELF_MP_CS_CCP_RGT_STOP	Notification of CCP RGT stop
1116	CELF_MP_CS_CCP_HRGT_START	Start notification of incoming of a CCP hold call
1117	CELF_MP_CS_CCP_HRGT_STOP	Stop notification of incoming of a CCP hold call
1118	CELF_MP_CS_CCP_DST_START	Notification of CCP DST start
1119	CELF_MP_CS_CCP_DST_STOP	Notification of CCP DST stop
1120	CELF_MP_CS_CCP_RBT_START	Notification of CCP RBT start
1121	CELF_MP_CS_CCP_RBT_STOP	Notification of CCP RBT stop
1122	CELF_MP_CS_CCP_BT_START	Notification of CCP BT start

1123	CELF_MP_CS_CCP_CWT_START	Notification of CCP CWT start
1124	CELF_MP_CS_CCP_CWT_STOP	Notification of CCP CWT stop
1125	CELF_MP_CS_CCP_REJECT_ASK	Inquiry report of rejecting a CCP CS incoming call
1126		

### 1127 1.4.3 Notification type

1128	CELF_MP_CS_RSMP_REST_STA:	Restriction display start notification
1129	CELF_MP_CS_RSMP_REST_END:	Restriction display end notification
1130		

### 1131 1.4.4 Restriction status

1132 The 0th bit is used for PS restriction status, and the 1st bit is used for CS restriction status.  
1133 (Bit ON means "restricted." Bit OFF means "unrestricted.")

1134	CELF_MP_CS_BIT_RESTINF_CS:	CS restriction information
1135	CELF_MP_CS_BIT_RESTINF_PS:	PS restriction information

1136 The 2nd bit is used for PS emergency restriction status, and the 3rd bit is used for CS emergency  
1137 restriction status.

1138	CELF_MP_CS_BIT_ECRESTINF_CS:	Emergency CS restriction information
1139	CELF_MP_CS_BIT_ECRESTINF_PS:	Emergency PS restriction information

1140  
1141  
1142

1143

## 2. Start Notification

1144

### 2.1 Symbol: `celf_mp_cs_notification_start`

1145

#### 2.1.1 Syntax

1146

```
CelfMpStatus celf_mp_cs_notification_start (  
1147     CelfMpAppID      app_id,  
1148     CelfMpCsNotifySet event_set,  
1149     CelfMpCallback   callback_func);
```

1150

#### 2.1.2 Argument

1151

Name: `app_id`

1152

Type: `CelfMpAppId`

1153

I/O: `I`

1154

Description:

1155

Application identifier.

1156

1157

Name: `event_set`

1158

Type: `CelfMpCsNotifySet`

1159

I/O: `I`

1160

Description:

1161

Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class

1162

**may** be registered to have a callback function called when the event occurs for the application identified by

1163

`app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

1164

1165

The event classes are defined as follows:

1166

`CELF_MP_CS_CLASS_COM_STATUS`: Voice communication status notification

1167

`CELF_MP_CS_CLASS_TLK_TIME`: Call duration notification

1168

`CELF_MP_CS_CLASS_DISC_CAUSE`: Disconnection cause notification

1169

`CELF_MP_CS_CLASS_FW_RESULT`: Call forwarding result notification

1170

`CELF_MP_CS_CLASS_OFFHK_TO`: Off-hook originating timeout notification

1171

1172

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

1173

`CELF_MP_CS_CLASS_ALL`, however to reduce overhead it is recommended that only the needed event

1174

classes **should** be registered.

1175

1176

Name: `callback_func`

1177

Type: `CelfMpCallback`

1178

I/O: `I`

1179 Description:

1180 The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

1181

### 1182 2.1.3 Return Value

1183 Type: `CelfMpStatus`

1184 Description:

1185 `celf_mp_cs_notification_start()` **shall** return one of the following values:

1186 `CELFP_MP_STATUS_OK`: successful completion

1187 `CELFP_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1188 `CELFP_MP_STATUS_EVENT_SET_ERR`: Notification event set is not valid

1189 `CELFP_MP_STATUS_ERR`: Other unsuccessful completion.

1190

### 1191 2.1.4 Include File

1192 `/usr/include/celf/mp_cs.h`

1193

### 1194 2.1.5 Functional Description

1195 This function is used to start notification callbacks for events related to circuit switched communication.

1196 Events from a registered class **shall** cause the registered callback function to be called when the event  
1197 occurs for the application identified by `app_id`. If a class of events does not have a registered callback  
1198 function, no callback **shall** occur for those events.

1199

1200 The event structure in section 0.1.1 **must** be used and the value subtype **shall be set to**  
1201 **“VoiceNotify\_ConnInfo”**.

1202

1203

1204

## 3. Stop Notification

1205

### 3.1 Symbol: `self_mp_cs_notification_stop`

1206

#### 3.1.1 Syntax

1207

```
CelfMpStatus self_mp_cs_notification_stop (
```

1208

```
    CelfMpAppId  app_id,
```

1209

```
    CelfMpCsNotifySet  event_set);
```

1210

#### 3.1.2 Argument

1211

Name: `app_id`

1212

Type: `CelfMpAppId`

1213

I/O: `I`

1214

Description:

1215

Application identifier.

1216

1217

Name: `event_set`

1218

Type: `CelfMpCsNotifySet`

1219

I/O: `I`

1220

Description:

1221

Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class

1222

**may** be registered to have a callback function called when the event occurs for the application identified by

1223

`app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

1224

1225

The event classes are defined as follows:

1226

`SELF_MP_CS_CLASS_COM_STATUS`: Voice communication status notification

1227

`SELF_MP_CS_CLASS_TLK_TIME`: Call duration notification

1228

`SELF_MP_CS_CLASS_DISC_CAUSE`: Disconnection cause notification

1229

`SELF_MP_CS_CLASS_FW_RESULT`: Call forwarding result notification

1230

`SELF_MP_CS_CLASS_OFFHK_TO`: Off-hook originating timeout notification

1231

1232

#### 3.1.3 Return Value

1233

Type: `CelfMpStatus`

1234

Description:

1235

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

1236

`SELF_MP_STATUS_OK`: successful completion

1237

`SELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1238

`SELF_MP_STATUS_EVENT_SET_ERR`: Notification event set is not valid

1239 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1240

### 1241 3.1.4 Include File

1242 /usr/include/celf/mp\_cs.h

1243

### 1244 3.1.5 Functional Description

1245 This function stops voice communication related event reporting.

1246 For notification events, see "Start notification".

1247 Note: For further information about the event structure consult section 0.1 in this document.

1248

1249

1250

DRAFT

## 1251 4. Get Voice Communication Status

### 1252 4.1 Symbol: `celf_mp_cs_get_com_status`

#### 1253 4.1.1 Syntax

```
1254 CelfMpStatus celf_mp_cs_get_com_status (  
1255     CelfMpAppId  app_id);
```

#### 1256 4.1.2 Argument

1257 Name: `app_id`

1258 Type: `CelfMpAppId`

1259 I/O: `I`

1260 Description:

1261 Application identifier.

1262

#### 1263 4.1.3 Return Value

1264 Type: `CelfMpStatus`

1265 Description:

1266 `celf_mp_cs_get_com_status()` shall return one of the values defined in section 0.1.

1267

#### 1268 4.1.4 Include File

1269 `/usr/include/celf/mp_cs.h`

1270

#### 1271 4.1.5 Functional Description

1272 This function gets the current voice communication status.

1273 Without the monitoring the voice communication, it is possible to get the status of voice communication.

1274

1275

1276

## 1277 5. Get Connection Information to Other Party

### 1278 5.1 Symbol: `celf_mp_cs_get_con_info_ref`

#### 1279 5.1.1 Syntax

```
1280 CelfMpStatus celf_mp_cs_get_con_info_ref (  
1281     CelfMpAppId      app_id,  
1282     CelfMpCallNo     call_no,  
1283     CelfMpConnectInfo connect_inf_p);
```

#### 1284 5.1.2 Argument

1285 Name: `app_id`

1286 Type: `CelfMpAppId`

1287 I/O: `I`

1288 Description:

1289 Application identifier.

1290

1291 Name: `call_no`

1292 Type: `CelfMpCallNo`

1293 I/O: `I`

1294 Description:

1295 Call reference (0 to 255).

1296

1297 Name: `connect_inf_p`

1298 Type: `CelfMpConnectInfo`

1299

1300 Description:

1301 Pointer to the connection destination information. See section 0.1 for details.

1302

#### 1303 5.1.3 Return Value

1304 Type: `CelfMpStatus`

1305 Description:

1306 `celf_mp_cs_get_con_info_ref()` **shall** return one of the values defined:

1307 `CELF_MP_STATUS_OK`: successful completion

1308 `CELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1309 `CELF_MP_STATUS_CALL_NO_ERR`: Call number is not valid

1310 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

1311



1312 **5.1.4 Include File**

1313 /usr/include/celf/mp\_cs.h

1314

1315 **5.1.5 Functional Description**

1316 This function refers to the connection information to other party specified call reference

1317 Without the monitoring the voice communication, it is possible to get the connection information

1318

1319 In the following cases, The result (STS) is set CELF\_MP\_CS\_ERR.

1320 1. The call specified by call reference does not exist.

1321 2. Other parameter Error.

DRAFT

1322 **6. Get Call Duration**

1323 **6.1 Symbol: celf\_mp\_cs\_get\_call\_duration**

1324 **6.1.1 Syntax**

```
1325 CelfMpStatus celf_mp_cs_get_call_duration (  
1326     CelfMpAppId  app_id,  
1327     CelfMpTime   time);
```

1328 **6.1.2 Argument**

1329 Name: app\_id

1330 Type: CelfMpAppId

1331 I/O: I

1332 Description:

1333 Application identifier.

1334

1335 Name: time

1336 Type: CelfMpTime

1337 I/O: O

1338 Description:

1339 `celf_mp_cs_get_call_duration()` shall return the current call duration in seconds.

1340 **6.1.3 Return Value**

1341 Type: CelfMpStatus

1342 Description:

1343 `celf_mp_cs_get_con_info_ref()` shall return one of the values defined:

1344 **CELF\_MP\_STATUS\_OK:** successful completion

1345 **CELF\_MP\_STATUS\_APP\_ID\_ERR:** Application ID is not valid.

1346 **CELF\_MP\_STATUS\_CALL\_NO\_ERR:** Call number is not valid

1347 **CELF\_MP\_STATUS\_ERR:** Other unsuccessful completion.

1348

1349

1350 **6.1.4 Include File**

1351 `/usr/include/celf/mp_cs.h`

1352

1353 **6.1.5 Functional Description**

1354 This function gets the call duration on the current call.

1355 The call duration is counted by the voice communication service.

1356 When no call exists, the function returns zero.  
1357

DRAFT

## 1358 7. Off-Hook Notification

### 1359 7.1 Symbol: `celf_mp_cs_notification_off_hook`

#### 1360 7.1.1 Syntax

```
1361 CelfMpStatus celf_mp_cs_notification_off_hook (  
1362     CelfMpAppId  app_id,  
1363     CelfMpCsBtype com_type,  
1364     CelfMpCsOffHk option);
```

#### 1365 7.1.2 Argument

1366 Name: `app_id`

1367 Type: `CelfMpAppId`

1368 I/O: `I`

1369 Description:

1370 Application identifier.

1371

1372 Name: `com_type`

1373 Type: `celfCsBtype`

1374 I/O: `I`

1375 Description:

1376 Communication type as defined in section 0.1.

1377

1378 Name: `option`

1379 Type: `CelfMpCsOffHk`

1380 I/O: `I`

1381 Description:

1382 One the following options **shall** be set:

1383 `CELF_MP_CS_OFFHK_AUTO` Automatic transmission

1384 `CELF_MP_CS_OFFHK_MANUAL` Manual transmission

1385

#### 1386 7.1.3 Return Value

1387 Type: `CelfMpStatus`

1388 Description:

1389 `celf_mp_cs_notification_off_hook()` **shall** return one of the values defined:

1390 `CELF_MP_STATUS_OK`: successful completion

1391 `CELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1392 `CELF_MP_STATUS_COM_TYPE_ERR`: Communication type is not valid

1393 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1394

## 1395 7.1.4 Include File

1396 /usr/include/celf/mp\_cs.h

1397

## 1398 7.1.5 Functional Description

1399 This function receives the request of off-hook.

1400

1401 The term “off-hook” refers to the user first presses the "dial" button, then enters the number to dial.

1402

1403 By this function,

1404 (1) When the mobile phone is in the wait (standby) status, the dial tone (DT) sounds and it is possible to  
1405 input dial number, or

1406 (2) When the input of dial number is completed, the mobile phone starts the originating.

1407 Because the function is an immediate return function, to confirm the complete result, including the  
1408 negotiation with the network, `celf_mp_cs_notification_status()` shall be used to obtain the  
1409 communication status.

1410

1411 The process at timer timeout (five seconds) varies depending on the specification of “option”.

1412 This timer count starts at the last dial inputting.

1413 (1) When the "option" is `CELF_MP_CS_OFFHK_AUTO` (automatic originating)

1414 Automatic originating operation is immediately performed by the dials, which were already input in  
1415 "Dial ".

1416 (2) When the "option" is `CELF_MP_CS_OFFHK_MANUAL` (manual originating)

1417 It is notified timeout to an application, and waits for the notification of originating from  
1418 the application. ("Complete dial" or "On-hook originating")

1419 Timeout is notified by monitoring "Off-hook originating timeout notification" in "Start  
1420 voice communication status monitoring".

1421

1422 When a mobile phone is moved to low voltage mode, a low voltage notification is sent.

1423 During low voltage, when the communication status is other than the under standby, this Off-hook is  
1424 disabled.

1425

1426 If an incoming call arrives during off-hook, this Off-hook is cancelled.

1427

1428 In case of using the subaddress, it should be use the function "On-hook originating".

1429

## 8. Disconnect

1430

### 8.1 Symbol: `self_mp_cs_disconnect`

1431

#### 8.1.1 Syntax

1432

```
CelfMpStatus self_mp_cs_disconnect (
```

1433

```
    CelfMpAppId  app_id
```

1434

```
    CelfMpCsBtype com_type);
```

1435

#### 8.1.2 Argument

1436

Name: `app_id`

1437

Type: `CelfMpAppId`

1438

I/O: `I`

1439

Description:

1440

Application identifier.

1441

1442

Name: `com_type`

1443

Type: `CelfMpCsBtype`

1444

I/O: `I`

1445

Description:

1446

Communication type as defined in section 0.1.

1447

1448

#### 8.1.3 Return Value

1449

Type: `CelfMpStatus`

1450

Description:

1451

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

1452

`SELF_MP_STATUS_OK`: successful completion

1453

`SELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1454

`SELF_MP_STATUS_COM_TYPE_ERR`: Communication type is not valid

1455

`SELF_MP_STATUS_ERR`: Other unsuccessful completion.

1456

1457

#### 8.1.4 Include File

1458

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

1459

1460

#### 8.1.5 Functional Description

1461

This function receives the request to disconnect the call.

1462

**Classification: *Circuit Switched Service***

- 1463 Because the function is an immediate return function, to confirm the complete result, including the  
1464 negotiation with the network, it should be issued “celf\_mp\_cs\_notification\_status()” to obtain the  
1465 communication status.
- 1466
- 1467 An incoming call cannot be disconnected by this function. (Use "Reject incoming call")
- 1468
- 1469 If multiple calls exist, all calls are disconnected.

DRAFT

1470 **9. Dial**

1471 **9.1 Symbol: celf\_mp\_cs\_dial**

1472 **9.1.1 Syntax**

```
1473 CelfMpStatus celf_mp_cs_dial (  
1474     CelfMpAppId  app_id  
1475     CelfMpCsBtype com_type,  
1476     CelfMpCsDialBuffer  dial_buf,  
1477     CelfMpCsDialLen    dial_len);
```

1478 **9.1.2 Argument**

1479 Name: app\_id

1480 Type: CelfMpAppId

1481 I/O: I

1482 Description:

1483 Application identifier.

1484

1485 Name: com\_type

1486 Type: CelfMpCsBtype

1487 I/O: I

1488 Description:

1489 Communication type as defined in section 0.1.

1490

1491 Name: dial\_buf

1492 Type: CelfMpCsDialBuffer

1493 I/O: I

1494 Description:

1495 Dial data buffer address

1496

1497 Name: dial\_len

1498 Type: CelfMpCsDialLen

1499 I/O: I

1500 Description:

1501 Dial data length

1502

1503 **9.1.3 Return Value**

1504 Type: CelfMpStatus



1505 Description:

1506 `celf_mp_cs_dial()` shall return one of the values defined:

1507 CELF\_MP\_STATUS\_OK: successful completion

1508 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

1509 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid

1510 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1511

## 1512 9.1.4 Include File

1513 `/usr/include/celf/mp_cs.h`

1514

## 1515 9.1.5 Functional Description

1516 This function receives the sequence of dial number.

1517

1518 Because the function is an immediate return function, to confirm the complete result, including the  
1519 negotiation with the network, it should be issued "`celf_mp_cs_notification_status()`" to obtain the  
1520 communication status.

1521

1522 The dial data stores the following ASCII codes.

1523 1 : 0 x 31    2 : 0 x 32    3 : 0 x 33

1524 4 : 0 x 34    5 : 0 x 35    6 : 0 x 36

1525 7 : 0 x 37    8 : 0 x 38    9 : 0 x 39

1526 \* : 0 x 2a    0 : 0 x 30    # : 0 x 23

1527

1528 Under this off-hook status, the mobile phone starts an outgoing call with "Dial" and "Complete dial".

1529 Five seconds later from the last digit has been entered, the outgoing process starts automatically, when  
1530 automatic transmission is specified in "Off-hook".

1531 When "Off-hook" is called, the mobile phone is in off-hook status.

1532

1533 Under this on-hook status, DTMF is sent, if the status is (a) the conversation or (b) the conversation and  
1534 hold.

1535 **10.Dial Complete**

1536 **10.1 Symbol: celf\_mp\_cs\_dial\_end**

1537 **10.1.1 Syntax**

```
1538 CelfMpStatus celf_mp_cs_dial_end (  
1539     CelfMpAppId  app_id  
1540     CelfMpCsBtype com_type);
```

1541 **10.1.2 Argument**

1542 Name: app\_id

1543 Type: CelfMpAppId

1544 I/O: I

1545 Description:

1546 Application identifier.

1547

1548 Name: com\_type

1549 Type: CelfMpCsBtype

1550 I/O: I

1551 Description:

1552 Communication type as defined in section 0.1.

1553

1554 **10.1.3 Return Value**

1555 Type: CelfMpStatus

1556 Description:

1557 `celf_mp_cs_dial_end()` shall return one of the values defined:

1558 CELF\_MP\_STATUS\_OK: successful completion

1559 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

1560 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid

1561 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1562

1563 **10.1.4 Include File**

1564 `/usr/include/celf/mp_cs.h`

1565

1566 **10.1.5 Functional Description**

1567 This function receives the request to end the dial entry.

1568

1569 Because this is an asynchronous function the service will return the result through a notification.

1570 `celf_mp_cs_notification_status()` shall be used to obtain the communication status.

1571 Under off-hook status, the mobile phone starts outgoing operation by calling this function with dial  
1572 number, which was given by preceding function calls "Dial".

1573

1574 Under on-hook status, the calling this function is disabled.

1575

DRAFT

1576 **11.Response to Incoming Call**

1577 **11.1 Symbol: celf\_mp\_cs\_call\_rcv**

1578 **11.1.1 Syntax**

1579 CelfMpStatus celf\_mp\_cs\_call\_rcv (  
1580           CelfMpAppId   app\_id  
1581           CelfMpCsBtype com\_type);

1582 **11.1.2 Argument**

1583 Name: app\_id

1584 Type: CelfMpAppId

1585 I/O: I

1586 Description:

1587 Application identifier.

1588

1589 Name: com\_type

1590 Type: CelfMpCsBtype

1591 I/O: I

1592 Description:

1593 Communication type as defined in section 0.1.

1594

1595 **11.1.3 Return Value**

1596 Type: CelfMpStatus

1597 Description:

1598 `celf_mp_cs_call_rcv()` shall return one of the values defined:

1599 CELF\_MP\_STATUS\_OK: successful completion

1600 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

1601 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid

1602 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1603

1604 **11.1.4 Include File**

1605 `/usr/include/celf/mp_cs.h`

1606

1607 **11.1.5 Functional Description**

1608 This function receives the request to process an incoming call.

1609

Classification: *Circuit Switched Service*

1610 Because the function is an immediate return function, to confirm the complete result, including the  
1611 negotiation with the network, it should be issued “celf\_mp\_cs\_notification\_status()” to obtain the  
1612 communication status.

1613

1614 One of the following operations is performed depending on the mobile phone status.

1615 Under incoming : Responds to the incoming call.

1616 Under response hold : Responds to the response hold call

1617 Others : Disabled

1618

1619 If the mobile phone is in low voltage mode, this function is disabled.

1620

1621 To respond to the incoming call in the status, “under conversation and incomings”, use "Reject incoming  
1622 call".

DRAFT

1623 **12.Forward Incoming Call**

1624 **12.1 Symbol: celf\_mp\_cs\_call\_forward**

1625 **12.1.1 Syntax**

1626 CelfMpStatus celf\_mp\_cs\_call\_forward (  
1627 CelfMpCsBtype com\_type);

1628 **12.1.2 Argument**

1629

1630 Name: com\_type

1631 Type: CelfMpCsBtype

1632 I/O: I

1633 Description:

1634 Communication type as defined in section 0.1.

1635

1636 **12.1.3 Return Value**

1637 Type: CelfMpStatus

1638 Description:

1639 celf\_mp\_cs\_call\_forward() **shall** return one of the values defined:

1640 CELF\_MP\_STATUS\_OK: successful completion

1641 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid

1642 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1643

1644 **12.1.4 Include File**

1645 /usr/include/celf/mp\_cs.h

1646

1647 **12.1.5 Functional Description**

1648 This function receives the request to forward an incoming call.

1649

1650 Because the function is an immediate return function, to confirm the complete result, including the  
1651 negotiation with the network, a “celf\_mp\_cs\_notification\_status()” should be issued to obtain the  
1652 communication status.

1653

1654 The incoming call is forwarded when the communication status is (a)under the incoming, (b)under  
1655 conversation and incoming, or (c)under hold and incoming.

1656

1657 If the forwarding fails, incoming call is continued between other party and this phone.

DRAFT

1659 **13.Forward to Voice Mail System**

1660 **13.1 Symbol: celf\_mp\_cs\_call\_forward\_voice\_msg**

1661 **13.1.1 Syntax**

1662 CelfMpStatus celf\_mp\_cs\_call\_forward\_voice\_msg (  
1663 CelfMpCsBtype com\_type);

1664 **13.1.2 Argument**

1665  
1666 Name: com\_type  
1667 Type: CelfMpCsBtype  
1668 I/O: I  
1669 Description:  
1670 Communication type as defined in section 0.1.  
1671

1672 **13.1.3 Return Value**

1673 Type: CelfMpStatus  
1674 Description:  
1675 celf\_mp\_cs\_call\_forward\_voice\_msg() **shall** return one of the values defined:  
1676 CELF\_MP\_STATUS\_OK: successful completion  
1677 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid  
1678 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.  
1679

1680 **13.1.4 Include File**

1681 /usr/include/celf/mp\_cs.h

1683 **13.1.5 Functional Description**

1684 This function receives the request to forward a call to a voice mail system.

1685  
1686 Because the function is an immediate return function, to confirm the complete result, including the  
1687 negotiation with the network, it should be issued “celf\_mp\_cs\_notification\_status()” to obtain the  
1688 communication status.

1689  
1690 The incoming call is forwarded to phone-answering message when the communication status is (a) under  
1691 the incoming, (b) under conversation and incoming, or (c) under hold and incoming.

1692  
1693 If the forwarding fails, incoming call is continued between other party and this phone.



1694

## 14.Call Hold

1695

### 14.1 Symbol: `celf_mp_cs_call_hold`

1696

#### 14.1.1 Syntax

1697

`CelfMpStatus celf_mp_cs_call_hold (`

1698

`CelfMpCsBtype com_type);`

1699

#### 14.1.2 Argument

1700

1701 Name: `com_type`

1702 Type: `CelfMpCsBtype`

1703 I/O: `I`

1704 Description:

1705 Communication type as defined in section 0.1.

1706

#### 14.1.3 Return Value

1708 Type: `CelfMpStatus`

1709 Description:

1710 `celf_mp_cs_call_hold()` shall return one of the values defined:

1711 `CELF_MP_STATUS_OK:` successful completion

1712 `CELF_MP_STATUS_COM_TYPE_ERR:` Communication type is not valid

1713 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

1714

#### 14.1.4 Include File

1716 `/usr/include/celf/mp_cs.h`

1717

#### 14.1.5 Functional Description

1719 This function receives the requests response hold.

1720

1721 Because the function is an immediate return function, to confirm the complete result, including the  
1722 negotiation with the network, it should be issued "`celf_mp_cs_notification_status()`" to obtain the  
1723 communication status.

1724

1725 This response hold is performed for an incoming call, only when the communication status is under  
1726 incoming.

1727

1728 To release response hold (move to the under conversation status) call "Response to an incoming call".

1729

1730

DRAFT

1731

## 15.Call Reject

1732

### 15.1 Symbol: `celf_mp_cs_call_reject`

1733

#### 15.1.1 Syntax

1734

`CelfMpStatus celf_mp_cs_call_reject (`

1735

`CelfMpCsBtype com_type);`

1736

#### 15.1.2 Argument

1737

1738

Name: `com_type`

1739

Type: `CelfMpCsBtype`

1740

I/O: `I`

1741

Description:

1742

Communication type as defined in section 0.1.

1743

1744

#### 15.1.3 Return Value

1745

Type: `CelfMpStatus`

1746

Description:

1747

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

1748

`CELF_MP_STATUS_OK:` successful completion

1749

`CELF_MP_STATUS_COM_TYPE_ERR:` Communication type is not valid

1750

`CELF_MP_STATUS_ERR:` Other unsuccessful completion.

1751

1752

#### 15.1.4 Include File

1753

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

1754

1755

#### 15.1.5 Functional Description

1756

This function receives the request to reject an incoming call.

1757

1758

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

1759

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

1760

communication status.

1761

1762

The operation for each communication status is as follows:

1763

Under incoming:                      Rejects an incoming call

1764

Under conversation and incoming: Rejects an incoming call

1765

Under hold and incoming:            Rejects an incoming call

1766 Under conversation, hold, and incoming: Rejects an incoming call  
1767  
1768  
1769

DRAFT

1770 **16.Multi Party Call**

1771 **16.1 Symbol: celf\_mp\_cs\_mp\_call**

1772 **16.1.1 Syntax**

1773 CelfMpStatus celf\_mp\_cs\_mp\_call (  
1774       CelfMpCsBtype com\_type,  
1775       CelfMpCsMop mode,  
1776       CelfMpCallRef call\_reference);

1777 **16.1.2 Argument**

1778

1779 Name: com\_type

1780 Type: CelfMpCsBtype

1781 I/O: I

1782 Description:

1783 Communication type as defined in section 0.1.

1784

1785 Name: mode

1786 Type: CelfMpCsMop

1787 I/O: I

1788 Description:

1789 Operation type

1790 CELF\_MP\_CS\_MOP\_RSV\_DISC: Disconnect the hold call

1791 CELF\_MP\_CS\_MOP\_DISC\_AND\_RSP: Response after disconnection

1792 CELF\_MP\_CS\_MOP\_RSV\_AND\_RSP: Response after hold (including operation for switching a call)

1793 CELF\_MP\_CS\_MOP\_CR\_DISC: Disconnect call specified by the call reference

1794

1795 Name: call\_reference

1796 Type: CelfMpCallRef

1797 I/O: I

1798 Description:

1799 Call reference of the call to be disconnected

1800 Valid only if CELF\_MP\_CS\_MOP\_CR\_DISC is specified for the second argument.

1801

1802

1803 **16.1.3 Return Value**

1804 Type: CelfMpStatus

1805 Description:

1806 `celf_mp_cs_mp_call()` shall return one of the values defined:

1807 CELF\_MP\_STATUS\_OK: successful completion

1808 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid

1809 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1810

## 1811 16.1.4 Include File

1812 `/usr/include/celf/mp_cs.h`

1813

## 1814 16.1.5 Functional Description

1815 This function receives the request to operate for each call, when communication is made with multiple  
1816 calls.

1817

1818 The operation is as follows depending on CELF\_MP\_CS\_MOP:

1819 - CELF\_MP\_CS\_MOP\_RSV\_DISC

1820 If a hold call exists, this hold call is disconnected.

1821

1822 - CELF\_MP\_CS\_MOP\_DISC\_AND\_RSP

1823 If a conversation call exists and if another call status is incoming or hold, the conversation call transits to  
1824 disconnect status and another call transits to conversation status.

1825 See detail below.

1826 (1) Under conversation and incoming

1827 This status is that 1st call is in conversation, and 2nd call is incoming.

1828 The result is that 1st call is released, and 2nd call is conversation.

1829 (2) Under conversation and hold

1830 This status is that 1st call is in conversation, and 2nd call is hold.

1831 The result is that 1st call is released, and 2nd call is conversation.

1832 (3) Under conversation, hold, and incoming

1833 This status is that 1st call is in conversation, that 2nd call is hold, and that 3rd call is incoming.

1834 The result is that 1st call is released, that 2nd call maintains hold, and that 3rd call is conversation.

1835 (4) Under response hold

1836 This status is not changed.

1837

1838 - CELF\_MP\_CS\_MOP\_RSV\_AND\_RSP

1839 If a conversation call exists and if another call status is incoming or hold,

1840 the conversation call transits to hold status and another call transits to conversation status.

1841 See detail below.

Classification: *Circuit Switched Service*

1842 (1) Under conversation and incoming  
1843 This status is that 1st call is in conversation, and 2nd call is incoming.  
1844 The result is that 1st call is hold, and 2nd call is conversation.  
1845 (2) Under conversation and hold  
1846 This status is that 1st call is in conversation, and 2nd call is hold.  
1847 The result is that 1st call is hold, and 2nd call is conversation.  
1848 (3) Under conversation, hold, and incoming  
1849 This status is that 1st call is in conversation, that 2nd call is hold, and that 3rd call is incoming.  
1850 The result is that 1st call is hold, that 2nd call is in conversation, that 3rd call maintains incoming.  
1851 (4) Under response hold  
1852 This status is that 1st call is hold.  
1853 The result is that 1st call is in conversation.  
1854  
1855 - CELF\_MP\_CS\_MOP\_CR\_DISC It is disconnect the call specified the call reference.  
1856  
1857 Because the function is an immediate return function, to confirm the complete result, including the  
1858 negotiation with the network, it should be issued "celf\_mp\_cs\_notification\_start()" to obtain the  
1859 communication status.

1860 **17.On-Hook Originating**

1861 **17.1 Symbol: celf\_mp\_cs\_originating\_on\_hook**

1862 **17.1.1 Syntax**

```
1863 CelfMpStatus celf_mp_cs_originating_on_hook (  
1864     CelfMpAppId      app_id,  
1865     CelfMpCsConReq   con_req);
```

1866 **17.1.2 Argument**

1867

1868 Name: app\_id

1869 Type: CelfMpAppId

1870 I/O: I

1871 Description:

1872 Application identifier.

1873

1874 Name: con\_req

1875 Type: CelfMpCsConReq

1876 I/O: I

1877 Description:

1878 Communication request type as defined in section 0.1.

1879

1880 **17.1.3 Return Value**

1881 Type: CelfMpStatus

1882 Description:

1883 celf\_mp\_cs\_call\_reject() **shall** return one of the values defined:

1884 CELF\_MP\_STATUS\_OK: successful completion

1885 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1886 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

1887 CELF\_MP\_CS\_ONHOOK\_DENY: On-hook originating is impossible.

1888 CELF\_MP\_CS\_ONHOOK\_STATUS\_ERR: Error due to communication conflict

1889 CELF\_MP\_CS\_ONHOOK\_OB\_CR: Excess of the maximum number of calls

1890

1891 **17.1.4 Include File**

1892 /usr/include/celf/mp\_cs.h

1893



1894 **17.1.5 Functional Description**

1895 This function receives the request to start an outgoing call with the specified dial number.

1896 The communication status should be Standby.

1897

1898 The dial number is specified by "dial\_buf" and "subaddr\_buf" in the "con\_req" structure.

1899

1900 If the character string, "184" or "186", is placed at the head of dial data, this character string is deleted.

1901 Whether the originating dial number is notified or not, it is identified by "notice".

1902

1903 The dial data and subaddress stores the following ASCII codes.

1904 1 : 0 x 31    2 : 0 x 32    3 : 0 x 33

1905 4 : 0 x 34    5 : 0 x 35    6 : 0 x 36

1906 7 : 0 x 37    8 : 0 x 38    9 : 0 x 39

1907 \* : 0 x 2a    0 : 0 x 30    # : 0 x 23

1908

1909 Because the function is an immediate return function, to confirm the complete result, including the  
1910 negotiation with the network, it should be issued "celf\_mp\_cs\_notification\_start()" to obtain the  
1911 communication status.

1912

1913 The originating request during low voltage is disabled.

1914 **18.Get Call Reference**

1915 **18.1 Symbol: celf\_mp\_cs\_get\_call\_reference**

1916 **18.1.1 Syntax**

```
1917 CelfMpStatus celf_mp_cs_get_call_reference (  
1918     CelfMpAppId  app_id  
1919     CelfMpCsChanNum  channel_num);  
1920
```

1921 **18.1.2 Argument**

1922 Name: app\_id

1923 Type: CelfMpAppId

1924 I/O: I

1925 Description:

1926 Application identifier.

1927

1928 Name: channel\_num

1929 Type: CelfMpCsChanNum

1930 I/O: O

1931 Description:

1932 Channel number information as defined in section 0.1.

1933

1934 **18.1.3 Return Value**

1935 Type: CelfMpStatus

1936 Description:

1937 `celf_mp_cs_get_call_reference()` **shall** return one of the values defined:

1938 CELF\_MP\_STATUS\_OK: successful completion

1939 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

1940 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

1941

1942 **18.1.4 Include File**

1943 `/usr/include/celf/mp_cs.h`

1944

1945 **18.1.5 Functional Description**

1946 This function gets the call reference in use.

1947

**Classification: *Circuit Switched Service***

1948 A value within 0 to 255 is set to "ChanNum\_00", "ChanNum\_01" and "ChanNum\_02". If channel is not  
1949 used, CELF\_MP\_CS\_CHAN\_NOUSE is set as the call reference.  
1950  
1951 Three channels correspond to three calls in multiple calls.  
1952

DRAFT

1953 **19.Start DCF message notification**

1954 **19.1 Symbol: celf\_mp\_cs\_DCF\_notification\_start**

1955 **19.1.1 Syntax**

```
1956 CelfMpStatus celf_mp_cs_DCF_notification_start (  
1957     CelfMpAppId  app_id,  
1958     CelfMpDCFSet event_set,  
1959     CelfMpCallback callback_func);
```

1960 **19.1.2 Argument**

1961 Name: app\_id

1962 Type: CelfMpAppId

1963 I/O: I

1964 Description:

1965 Application identifier.

1966

1967 Name: event\_set

1968 Type: CelfMpCsDCFSet

1969 I/O: I

1970 Description:

1971 Notification event set. Events that are classified as belonging to one of the CelfMpCsDCFSet class **may**  
1972 be registered to have a callback function called when the event occurs for the application identified by  
1973 app\_id. Classes of events are enabled by setting the corresponding bit in event\_set:

1974

1975 The event classes are defined as follows:

1976 CELF\_MP\_CS\_DCF\_DISP Display-related message

1977 CELF\_MP\_CS\_DCF\_HISTORY History-related message

1978 CELF\_MP\_CS\_DCF\_TONE1 Tone 1-related message

1979 CELF\_MP\_CS\_DCF\_TONE2 Tone 2-related message

1980 CELF\_MP\_CS\_DCF\_ETC Other messages

1981 CELF\_MP\_CS\_CLASS\_ALL All notified

1982

1983 A callback **may** be registered for all classes of events using special event class

1984 CELF\_MP\_CS\_CLASS\_ALL, however to reduce overhead it is recommended that only the needed event  
1985 classes **should** be registered.

1986

1987 Name: callback\_func

1988 Type: CelfMpCallback

1989 I/O: I

1990 Description:

1991 The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

### 1992 19.1.3 Return Value

1993 Type: `CelfMpStatus`

1994 Description:

1995 `celf_mp_cs_DCF_notification_start ()` **shall** return one of the values defined:

1996 `CELF_MP_STATUS_OK`: successful completion

1997 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

1998 `CELF_MP_STATUS_APP_ID_ERR`: Application ID is not valid.

1999

### 2000 19.1.4 Include File

2001 `/usr/include/celf/mp_cs.h`

2002

### 2003 19.1.5 Functional Description

2004 This function starts the monitoring the DCF message on the voice communication or AV communication.

2005

2006 The occurrence of the event is notified to the application, specified by `app_id`.

2007

2008 The messages to be notified are described below.

2009

2010 Display-related message:

2011 -Notification of starting display during CCP outgoing

2012 -Notification of starting display during CCP incoming

2013 -Notification of starting display during CCP calling

2014 -Notification of starting display during CCP connecting

2015 -Notification of starting display during CCP communication

2016 -Notification of ending CCP That is to notifies of release of a CCP call.

2017 -Notification of starting CCP disconnection (on the mobilephone) display

2018 -Notification of starting display of CCP disconnection (on the network) display

2019 -Notification of rejecting CCP outgoing

2020 -Notification of CCP hold

2021 -Notification of releasing CCP hold

2022

2023 History-related message:

2024 -Notification of registering CCP outgoing call history

**CE Linux Forum Technical Document**

**Classification: *Circuit Switched Service***

- 2025 -Notification of registering CCP absence incoming call history
- 2026 -Notification of registering CCP incoming call history
- 2027
- 2028 Tone 1-related message:(Tone sounding on the AP layer)
- 2029 -Notification of CCP RGT start
- 2030 -Notification of CCP RGT stop
- 2031 -Start report of incoming of a CCP hold call
- 2032 -Stop report of incoming of a CCP hold call
- 2033
- 2034 Tone 2-related message:(Tone sounding by the voice communication service)
- 2035 -Notification of CCP DST start
- 2036 -Notification of CCP DST stop
- 2037 -Notification of CCP RBT start
- 2038 -Notification of CCP RBT stop
- 2039 -Notification of CCP BT start
- 2040 -Notification of CCP CWT start
- 2041 -Notification of CCP CWT stop
- 2042
- 2043 Other messages:
- 2044 -Inquiry report of rejecting a CCP CS incoming call

**DRAFT**

2045 **20.Stop DCF message notification**

2046 **20.1 Symbol: celf\_mp\_cs\_DCF\_notification\_stop**

2047 **20.1.1 Syntax**

```
2048 CelfMpStatus celf_mp_cs_DCF_notification_stop (  
2049     CelfMpAppId  app_id  
2050     CelfMpDCFSet event_set);
```

2051 **20.1.2 Argument**

2052 Name: app\_id

2053 Type: CelfMpAppId

2054 I/O: I

2055 Description:

2056 Application identifier.

2057

2058 Name: event\_set

2059 Type: CelfMpCsDCFSet

2060 I/O: I

2061 Description:

2062 Notification event set. Events that are classified as belonging to one of the CelfMpCsDCFSet class.  
2063 Classes of events are enabled by setting the corresponding bit in event\_set:

2064

2065 The event classes are defined as follows:

2066 CELF\_MP\_CS\_DCF\_DISP Display-related message

2067 CELF\_MP\_CS\_DCF\_HISTORY History-related message

2068 CELF\_MP\_CS\_DCF\_TONE1 Tone 1-related message

2069 CELF\_MP\_CS\_DCF\_TONE2 Tone 2-related message

2070 CELF\_MP\_CS\_DCF\_ETC Other messages

2071 CELF\_MP\_CS\_CLASS\_ALL All notified

2072

2073

2074 **20.1.3 Return Value**

2075 Type: CelfMpStatus

2076 Description:

2077 celf\_mp\_cs\_DCF\_notification\_stop() **shall** return one of the values defined:

2078 CELF\_MP\_STATUS\_OK: successful completion

2079 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2080 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

2081

#### 2082 20.1.4 Include File

2083 /usr/include/celf/mp\_cs.h

2084

#### 2085 20.1.5 Functional Description

2086 This function stops notifying of the DCF message on voice communication or AV communication.

DRAFT



2087 **21.Voice Message Notification**

2088 **21.1 Symbol: celf\_mp\_cs\_voice\_msg\_notify**

2089 **21.1.1 Syntax**

2090 CelfMpStatus celf\_mp\_cs\_voice\_msg\_notify (  
2091 CelfMpCsRecMsg rec\_status);

2092 **21.1.2 Argument**

2093  
2094 Name: rec\_status  
2095 Type: CelfMpCsRecMsg  
2096 I/O: I  
2097 Description:  
2098 CELF\_MP\_CS\_REC\_MSG\_START: Start of a voice message  
2099 CELF\_MP\_CS\_REC\_MSG\_STOP: Stop of a voice message

2101 **21.1.3 Return Value**

2102 Type: CelfMpStatus  
2103 Description:  
2104 celf\_mp\_cs\_call\_voice\_msg\_notify() **shall** return one of the values defined:  
2105 CELF\_MP\_STATUS\_OK: successful completion  
2106 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid  
2107 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2109 **21.1.4 Include File**

2110 /usr/include/celf/mp\_cs.h

2112 **21.1.5 Functional Description**

2113 This function must be called before the communication state is changed to "under conversation."  
2114 After the start notification, a stop notification **must** be issued, when the voice message is stopped.

2115 **22.Hold Tone Start**

2116 **22.1 Symbol: celf\_mp\_cs\_hold\_tone\_start**

2117 **22.1.1 Syntax**

2118 CelfMpStatus celf\_mp\_cs\_hold\_tone\_start (  
2119 CelfMpAppId app\_id);

2120 **22.1.2 Argument**

2121  
2122 Name: app\_id  
2123 Type: CelfMpAppId  
2124 I/O: I  
2125 Description:  
2126 Application identifier.

2128 **22.1.3 Return Value**

2129 Type: CelfMpStatus  
2130 Description:  
2131 celf\_mp\_cs\_hold\_tone\_start() **shall** return one of the values defined:  
2132 CELF\_MP\_STATUS\_OK: successful completion  
2133 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid  
2134 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.  
2135 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

2137 **22.1.4 Include File**

2138 /usr/include/celf/mp\_cs.h

2140 **22.1.5 Functional Description**

2141 This function starts to sound a hold tone during a call.

2142 **23.Hold Tone Stop**

2143 **23.1 Symbol: celf\_mp\_cs\_hold\_tone\_stop**

2144 **23.1.1 Syntax**

2145 CelfMpStatus celf\_mp\_cs\_hold\_tone\_stop (  
2146 CelfMpAppId app\_id);

2147 **23.1.2 Argument**

2148  
2149 Name: app\_id  
2150 Type: CelfMpAppId  
2151 I/O: I  
2152 Description:  
2153 Application identifier.

2155 **23.1.3 Return Value**

2156 Type: CelfMpStatus  
2157 Description:  
2158 celf\_mp\_cs\_hold\_tone\_stop() **shall** return one of the values defined:  
2159 CELF\_MP\_STATUS\_OK: successful completion  
2160 CELF\_MP\_STATUS\_COM\_TYPE\_ERR: Communication type is not valid  
2161 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.  
2162 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.

2164 **23.1.4 Include File**

2165 /usr/include/celf/mp\_cs.h

2167 **23.1.5 Functional Description**

2168 This function stops to sound a hold tone during a call.

2169

## 2170 **24.Get 64K / AV Communication Status**

### 2171 **24.1 Symbol: celf\_mp\_cs\_get\_UD\_com\_stat**

#### 2172 **24.1.1 Syntax**

2173 CelfMpUDComStatus celf\_mp\_cs\_get\_UD\_com\_stat (  
2174 void);

#### 2175 **24.1.2 Argument**

2176 None.

2177

#### 2178 **24.1.3 Return Value**

2179 Type: CelfMpUDComStatus

2180 I/O: O

2181 Description:

2182 `celf_mp_cs_get_UD_com_stat()` **shall** return one of the values defined:

2183 CELF\_MP\_CS\_UD\_STOP: Under stop

2184 CELF\_MP\_CS\_UD\_RUN: Under communication

2185 CELF\_MP\_CS\_UD\_CALLED: Under incoming

2186 CELF\_MP\_CS\_UD\_CALLING: Under outgoing

2187 CELF\_MP\_CS\_UD\_DISCONNECT: Under disconnection

2188 CELF\_MP\_CS\_UD\_CALLING\_ALERT: Under calling

2189 CELF\_MP\_CS\_UD\_HOLD: Under hold

2190 CELF\_MP\_CS\_UD\_ERR: Error in UD Communication

2191

2192

#### 2193 **24.1.4 Include File**

2194 `/usr/include/celf/mp_cs.h`

2195

#### 2196 **24.1.5 Functional Description**

2197 This function refers to the communication status of 64K communication or AV communication.

2198

2199

## 2200 **25. Get internal/external AV Communication Status**

### 2201 **25.1 Symbol: celf\_mp\_cs\_get\_AV\_com\_stat**

#### 2202 **25.1.1 Syntax**

2203 CelfMpAVComStatus celf\_mp\_cs\_get\_AV\_com\_stat (  
2204 void);

#### 2205 **25.1.2 Argument**

2206 None.

2207

#### 2208 **25.1.3 Return Value**

2209 Type: CelfMpAVComStatus

2210 I/O: O

2211 Description:

2212 `celf_mp_cs_get_AV_com_stat()` **shall** return one of the values defined:

2213 CELF\_MP\_CS\_AV\_IN\_STOP: Under stop

2214 CELF\_MP\_CS\_AV\_IN\_RUN: Under communication

2215 CELF\_MP\_CS\_AV\_IN\_CALLED: Under incoming

2216 CELF\_MP\_CS\_AV\_IN\_CALLING: Under outgoing

2217 CELF\_MP\_CS\_AV\_IN\_DISCONNECT: Under disconnection

2218 CELF\_MP\_CS\_AV\_IN\_CALLING\_ALERT: Under calling

2219 CELF\_MP\_CS\_UD\_IN\_HOLD: Under hold

2220 CELF\_MP\_CS\_AV\_OUT\_STOP: Under stop

2221 CELF\_MP\_CS\_AV\_OUT\_RUN: Under communication

2222 CELF\_MP\_CS\_AV\_OUT\_CALLED: Under incoming

2223 CELF\_MP\_CS\_AV\_OUT\_CALLING: Under outgoing

2224 CELF\_MP\_CS\_AV\_OUT\_DISCONNECT: Under disconnection

2225 CELF\_MP\_CS\_AV\_OUT\_CALLING\_ALERT: Under calling

2226 CELF\_MP\_CS\_UD\_OUT\_HOLD: Under hold

2227

#### 2228 **25.1.4 Include File**

2229 `/usr/include/celf/mp_cs.h`

2230

#### 2231 **25.1.5 Functional Description**

2232 This function refers to the communication status of internal or external AV communication.

2233

## 26. Get Communication Status

2234

### 26.1 Symbol: `celf_mp_cs_get_com_stat`

2235

#### 26.1.1 Syntax

2236

`CelfMpCsComStatus celf_mp_cs_get_com_stat (`

2237

`CelfMpAppId        app_id,`

2238

`CelfMpCsRcvScene *  rcv_scene);`

2239

#### 26.1.2 Argument

2240

Name: `app_id`

2241

Type: `CelfMpAppId`

2242

I/O: `I`

2243

Description:

2244

Application identifier.

2245

Name: `rcv_scene`

2246

Type: `CelfMpCsRcvScene *`

2247

I/O: `O`

2248

Description:

2249

Incoming call type:

2250

`CELf_MP_CS_RCV_SCENE_COMPETE_TRN:` `Outgoing conflict`

2251

`CELf_MP_CS_RCV_SCENE_RSV_RETURN:` `Incoming hold call`

2252

`CELf_MP_CS_RCV_SCENE_CALL_BACK:` `Re-incoming`

2253

`CELf_MP_CS_RCV_SCENE_NORMAL:` `Normal`

2254

`CELf_MP_CS_RCV_SCENE_NON:` `Unset`

2255

2256

#### 26.1.3 Return Value

2257

Type: `CelfMpCsComStatus`

2258

Description:

2259

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

2260

Current communication status

2261

`CELf_MP_CS_COM_STATUS_WAIT:` `Standby`

2262

`CELf_MP_CS_COM_STATUS_RCV:` `Under incoming`

2263

`CELf_MP_CS_COM_STATUS_TRN:` `Under outgoing`

2264

`CELf_MP_CS_COM_STATUS_DLV:` `Under calling`

2265

`CELf_MP_CS_COM_STATUS_TLK:` `Under conversation`

2266

`CELf_MP_CS_COM_STATUS_HLD:` `Under response hold`

2267

`CELf_MP_CS_COM_STATUS_DUMMY1:` `Under off-hook`

Classification: *Circuit Switched Service*

2268 CELF\_MP\_CS\_COM\_STATUS\_RLS: Under release  
2269 CELF\_MP\_CS\_COM\_STATUS\_TLK\_RCV: Under conversation and incoming  
2270 CELF\_MP\_CS\_COM\_STATUS\_TLK\_TRN: Under conversation and outgoing  
2271 CELF\_MP\_CS\_COM\_STATUS\_TLK\_DLV: Under conversation and calling  
2272 CELF\_MP\_CS\_COM\_STATUS\_TLK\_RSV: Under conversation and hold  
2273 CELF\_MP\_CS\_COM\_STATUS\_TLK\_RLS: Under conversation and release  
2274 CELF\_MP\_CS\_COM\_STATUS\_TLK\_RSV\_RCV: Under conversation, hold, and incoming  
2275 CELF\_MP\_CS\_COM\_STATUS\_RCV\_AV: Under incoming of an AV call  
2276 CELF\_MP\_CS\_COM\_STATUS\_TRN\_AV: Under outgoing of an AV call  
2277 CELF\_MP\_CS\_COM\_STATUS\_DLV\_AV: Under calling of an AV call  
2278 CELF\_MP\_CS\_COM\_STATUS\_TLK\_AV: Under conversation of an AV call  
2279 CELF\_MP\_CS\_COM\_STATUS\_HLD\_AV: Under response hold of an AV call  
2280 CELF\_MP\_CS\_COM\_STATUS\_RLS\_AV: Under release of an AV call  
2281 CELF\_MP\_CS\_COM\_STATUS\_DUMMY2 : Under AV off-hook  
2282 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.  
2283 CELF\_MP\_CS\_ERR : Abnormal end  
2284

#### 26.1.4 Include File

2285 /usr/include/celf/mp\_cs.h  
2286

#### 26.1.5 Functional Description

2287  
2288  
2289 This function returns the incoming call status, when the current call is (a) under incoming status or (b)  
2290 under conversation and incoming status.  
2291

## 2292 27.Start Line Status Notification

### 2293 27.1 Symbol: `celf_mp_cs_line_status_notification_start`

#### 2294 27.1.1 Syntax

```
2295 CelfMpStatus celf_mp_cs_notification_start (  
2296     CelfMpAppId  app_id  
2297     CelfMpCsMtype event_set,  
2298     CelfMpCallback callback_func);
```

#### 2299 27.1.2 Argument

2300 Name: `app_id`

2301 Type: `CelfMpAppId`

2302 I/O: `I`

2303 Description:

2304 Application identifier.

2305

2306 Name: `event_set`

2307 Type: `CelfMpCsMtype`

2308 I/O: `I`

2309 Description:

2310 Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class  
2311 **may** be registered to have a callback function called when the event occurs for the application identified by  
2312 `app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

2313 `CELF_MP_CS_MONITOR_LINE_STATUS:` Line status change notification

2314 `CELF_MP_CS_MONITOR_RESTRICT:` Restriction status change notification

2315 `CELF_MP_CS_MONITOR_RSSI:` Receive level change notification

2316 `CELF_MP_CS_MONITOR_ALL:` All notified

2317

2318 Name: `callback_func`

2319 Type: `CelfMpCallback`

2320 I/O: `I`

2321 Description:

2322 The callback function, which **shall** be called when an event occurs from one of the classes in `event_set`.

2323

#### 2324 27.1.3 Return Value

2325 Type: `CelfMpStatus`

2326



2327 Description:

2328 `celf_mp_cs_notification_start()` **shall** return one of the values defined:

- 2329 CELF\_MP\_STATUS\_OK: successful completion  
2330 CELF\_MP\_STATUS\_APP\_ID\_ERR: Application ID is not valid.  
2331 CELF\_MP\_STATUS\_MON\_TYPE\_ERR: Monitor type is not valid  
2332 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2333

## 2334 27.1.4 Include File

2335 `/usr/include/celf/mp_cs.h`

2336

## 2337 27.1.5 Functional Description

2338 This function starts the monitoring the line status.

2339 The occurrence of the event is notified to the application, specified by `app_id`.

2340 The events to be notified are described below.

2341

2342 1. Line status change notification:

2343 This event notifies that the line status is changed.

2344 The line status is the out-of-communication area status and the within-communication area.

2345

2346 2. Restriction status change notification:

2347 This event notifies that a restriction status is changed.

2348 The restriction means that the incoming call or the outgoing call is restricted by the network in case of  
2349 traffic congestion.

2350

2351 3. Receive level change notification:

2352 This event notifies that the receive level is changed.

2353 The receive level is the intensity of electromagnetic wave. The intensity is four level, high, mid, low and  
2354 zero (out of area).

2355

2356 See section 0.1 for structure definitions and values.

2357

## 2358 28. Stop Line Status Notification

### 2359 28.1 Symbol: `celf_mp_cs_line_status_notification_stop`

#### 2360 28.1.1 Syntax

```
2361 CelfMpStatus celf_mp_cs_notification_stop (  
2362     CelfMpAppId  app_id  
2363     CelfMpCsMtype event_set);
```

#### 2364 28.1.2 Argument

2365 Name: `app_id`

2366 Type: `CelfMpAppId`

2367 I/O: `I`

2368 Description:

2369 Application identifier.

2370

2371 Name: `event_set`

2372 Type: `CelfMpCsMtype`

2373 I/O: `I`

2374 Description:

2375 Mask of the events for which reporting is to be stopped.

2376 Notification event set. Events that are classified as belonging to one of the `CelfMpCsNotifySet` class  
2377 **may** be registered to have a callback function called when the event occurs for the application identified by  
2378 `app_id`. Classes of events are enabled by setting the corresponding bit in `event_set`:

2379 `CELF_MP_CS_MONITOR_LINE_STATUS:` Line status change notification

2380 `CELF_MP_CS_MONITOR_RESTRICT:` Restriction status change notification

2381 `CELF_MP_CS_MONITOR_RSSI:` Received signal strength change notification

2382 `CELF_MP_CS_MONITOR_ALL:` All notified

2383

#### 2384 28.1.3 Return Value

2385 Type: `CelfMpStatus`

2386 Description:

2387 `celf_mp_cs_notification_stop()` **shall** return one of the values defined:

2388 `CELF_MP_STATUS_OK:` successful completion

2389 `CELF_MP_STATUS_APP_ID_ERR:` Application ID is not valid.

2390 `CELF_MP_STATUS_MON_TYPE_ERR:` Monitor type is not valid

2391 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2392

2393 **28.1.4 Include File**

2394 `/usr/include/celf/mp_cs.h`

2395

2396 **28.1.5 Functional Description**

2397 This function ends notifying on the event of the line status.

2398

DRAFT

2399 **29. Get Reception Level**

2400 **29.1 Symbol: celf\_mp\_cs\_get\_reception\_level**

2401 **29.1.1 Syntax**

2402 CelfMpReceptionLevel celf\_mp\_cs\_get\_reception\_level (  
2403 void);

2404 **29.1.2 Argument**

2405 None.

2406

2407 **29.1.3 Return Value**

2408 Type: CelfMpReceptionLevel

2409 I/O: O

2410 Description:

2411 `celf_mp_cs_get_reception_level()` **shall** return one of the values defined:

2412 CELF\_MP\_CS\_RSSI\_LEVEL\_0: Receive level 0

2413 CELF\_MP\_CS\_RSSI\_LEVEL\_1: Receive level 1

2414 CELF\_MP\_CS\_RSSI\_LEVEL\_2: Receive level 2

2415 CELF\_MP\_CS\_RSSI\_LEVEL\_3: Receive level 3

2416

2417 **29.1.4 Include File**

2418 `/usr/include/celf/mp_cs.h`

2419

2420 **29.1.5 Functional Description**

2421 This function obtains the current reception level.

2422 Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the  
2423 status of reception level.

2424

2425 **30. Get Line Status**

2426 **30.1 Symbol: celf\_mp\_cs\_get\_line\_status**

2427 **30.1.1 Syntax**

2428 CelfMpStatus celf\_mp\_cs\_get\_line\_status (  
2429           CELF\_MP\_CS\_AREAREF\_CHG\_INF \*           net);

2430 **30.1.2 Argument**

2431 Name: net

2432 Type: CELF\_MP\_CS\_AREAREF\_CHG\_INF

2433 I/O: I

2434 Description:

2435 Pointer to the struct used to hold line status information

2436

2437 **30.1.3 Return Value**

2438 Type: CelfMpStatus

2439 Description:

2440 celf\_mp\_cs\_get\_line\_status() shall return one of the values defined:

2441 CELF\_MP\_STATUS\_OK:           successful completion

2442 CELF\_MP\_STATUS\_ERR:       Other unsuccessful completion.

2443

2444 **30.1.4 Include File**

2445 /usr/include/celf/mp\_cs.h

2446

2447 **30.1.5 Functional Description**

2448 This function obtains the current line status.

2449 Without the line status monitoring by calling the “Start line status monitoring”, it is possible to get the  
2450 status of line status.

2451 See section 0.1 for further information.

2452

2453

## 31. Get Coverage Status

2454

### 31.1 Symbol: `celf_mp_cs_get_coverage_status`

2455

#### 31.1.1 Syntax

2456

`CelfMpStatus celf_mp_cs_get_line_status (`

2457

`CelfMpCsLineStatusEx* net,`

2458

`CelfMpCsCoverage cover);`

2459

#### 31.1.2 Argument

2460

Name: `net`

2461

Type: `CelfMpCsLineStatusEx`

2462

I/O: `I`

2463

Description:

2464

Pointer to the struct used to hold line status information

2465

2466

Name: `cover`

2467

Type: `CelfMpCsCoverage`

2468

I/O: `O`

2469

Description:

2470

Within- or out-of communication area status

2471

`CELF_MP_CS_LINE_STATUS_IN`: Within-communication area

2472

`CELF_MP_CS_LINE_STATUS_OUT`: Out-of-communication area

2473

2474

#### 31.1.3 Return Value

2475

Type: `CelfMpStatus`

2476

Description:

2477

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

2478

`CELF_MP_STATUS_OK`: successful completion

2479

`CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2480

2481

#### 31.1.4 Include File

2482

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

2483

2484

#### 31.1.5 Functional Description

2485

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

2487

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

2488 **32.Get Voice Mail Information**

2489 **32.1 Symbol: celf\_mp\_cs\_get\_vm\_info**

2490 **32.1.1 Syntax**

2491 CelfMpStatus celf\_mp\_cs\_get\_vm\_info (  
2492 CelfMpCsVMNum \* vm\_num);

2493 **32.1.2 Argument**

2494 Name: vm\_num

2495 Type: CelfMpCsVMNum

2496 I/O: I

2497 Description:

2498 Address of the storage area of the number of stored phone-answering messages

2499

2500 **32.1.3 Return Value**

2501 Type: CelfMpStatus

2502 Description:

2503 celf\_mp\_cs\_get\_vm\_info() **shall** return one of the values defined:

2504 CELF\_MP\_STATUS\_OK: successful completion

2505 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2506

2507 **32.1.4 Include File**

2508 /usr/include/celf/mp\_cs.h

2509

2510 **32.1.5 Functional Description**

2511 This function obtains the storage status of phone-answering messages from nonvolatile memory.

2512 The storage status is the number of message of phone-answering.

2513

2514

## 2515 **33.Set Voice Mail Information**

### 2516 **33.1 Symbol: celf\_mp\_cs\_set\_vm\_info**

#### 2517 **33.1.1 Syntax**

```
2518 CelfMpStatus celf_mp_cs_set_vm_info (  
2519     CelfMpCsVMNum    vm_num);
```

#### 2520 **33.1.2 Argument**

2521 Name: vm\_num

2522 Type: CelfMpCsVMNum

2523 I/O: I

2524 Description:

2525 The number of stored phone-answering messages

2526

#### 2527 **33.1.3 Return Value**

2528 Type: CelfMpStatus

2529 Description:

2530 `celf_mp_cs_set_vm_info()` **shall** return one of the values defined:

2531 CELF\_MP\_STATUS\_OK: successful completion

2532 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2533

#### 2534 **33.1.4 Include File**

2535 `/usr/include/celf/mp_cs.h`

2536

#### 2537 **33.1.5 Functional Description**

2538 This function sets the storage status of phone-answering message to non-volatile memory.

2539

2540

2541



2542 **34.Get Call Selection**

2543 **34.1 Symbol: celf\_mp\_cs\_get\_call\_select**

2544 **34.1.1 Syntax**

2545 CelfMpCallSelect celf\_mp\_cs\_get\_call\_select (  
2546 void);

2547 **34.1.2 Argument**

2548 None.

2549

2550 **34.1.3 Return Value**

2551 Type: CelfMpCallSelect

2552 I/O: O

2553 Description:

2554 CELF\_MP\_CS\_INCOMING\_VOICE\_ANSWERING Forward to the phone-answering message

2555 CELF\_MP\_CS\_INCOMING\_FORWARD Forward

2556 CELF\_MP\_CS\_INCOMING\_REJECT Reject (disconnect)

2557 CELF\_MP\_CS\_INCOMING\_NORMAL Receipt of an incoming call (normal incoming)

2558

2559 **34.1.4 Include File**

2560 /usr/include/celf/mp\_cs.h

2561

2562 **34.1.5 Functional Description**

2563 This function obtains the incoming call information from non-volatile memory.

2564 Refer "Set incoming function selection"

2565

2566

2567

## 35.Set Call Selection

2568

### 35.1 Symbol: `celf_mp_cs_set_call_select`

2569

#### 35.1.1 Syntax

2570

`CelfMpStatus celf_mp_cs_set_call_select (`

2571

`CelfMpCallSelect    select);`

2572

#### 35.1.2 Argument

2573

Name: `select`

2574

Type: `CelfMpCallSelect`

2575

I/O: `I`

2576

Description:

2577

`CELFP_MP_CS_INCOMING_VOICE_ANSWERING:` Forward to the phone-answering message

2578

`CELFP_MP_CS_INCOMING_FORWARD:` Forward

2579

`CELFP_MP_CS_INCOMING_REJECT:` Reject (disconnect)

2580

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

2581

2582

#### 35.1.3 Return Value

2583

Type: `CelfMpStatus`

2584

Description:

2585

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

2586

`CELFP_MP_STATUS_OK:` successful completion

2587

`CELFP_MP_STATUS_ERR:` Other unsuccessful completion.

2588

2589

#### 35.1.4 Include File

2590

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

2591

2592

#### 35.1.5 Functional Description

2593

This function sets the incoming call information to nonvolatile memory.

2594

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

2595

information.

2596

2597

## 2598 36.Set Service Information

### 2599 36.1 Symbol: `celf_mp_cs_set_service_info`

#### 2600 36.1.1 Syntax

```
2601 CelfMpStatus celf_mp_cs_set_service_info (  
2602     CelfMpRegNum     reg_no,  
2603     CelfMpCsSrvData * data);
```

#### 2604 36.1.2 Argument

2605 Name: `reg_no`

2606 Type: `CelfMpRegNum`

2607 I/O: `I`

2608 Description:

2609 Registration number: 1 to 10

2610 Name: `data`

2611 Type: `CelfMpCsSrvData`

2612 I/O: `I`

2613 Description:

2614 Pointer to supplementary service data

2615

#### 2616 36.1.3 Return Value

2617 Type: `CelfMpStatus`

2618 Description:

2619 `celf_mp_cs_set_service_info()` **shall** return one of the values defined:

2620 `CELF_MP_STATUS_OK:` successful completion

2621 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2622

#### 2623 36.1.4 Include File

2624 `/usr/include/celf/mp_cs.h`

2625

#### 2626 36.1.5 Functional Description

2627 This function registers the supplementary service information to the non-volatile memory,

2628

2629 The supplementary service information is the service name and Dial data for accessing the service.

2630 The `'reg_no'` is used as the key for accessing this supplementary service.

2631 The value range is from 0 to 10.

2632 See section 0.1 for additional information.

2633

DRAFT

## 2634 37. Get Service Information

### 2635 37.1 Symbol: `celf_mp_cs_get_service_info`

#### 2636 37.1.1 Syntax

```
2637 CelfMpStatus celf_mp_cs_get_service_info (  
2638         CelfMpRegNum      reg_no,  
2639         CelfMpCsSrvData *  data);
```

#### 2640 37.1.2 Argument

2641 Name: `reg_no`

2642 Type: `CelfMpRegNum`

2643 I/O: `I`

2644 Description:

2645 Registration number: 1 to 10

2646 Name: `data`

2647 Type: `CelfMpCsSrvData`

2648 I/O: `I`

2649 Description:

2650 Pointer to supplementary service data

2651

#### 2652 37.1.3 Return Value

2653 Type: `CelfMpStatus`

2654 Description:

2655 `celf_mp_cs_get_service_info()` **shall** return one of the values defined:

2656 `CELF_MP_STATUS_OK`: successful completion

2657 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2658

#### 2659 37.1.4 Include File

2660 `/usr/include/celf/mp_cs.h`

2661

#### 2662 37.1.5 Functional Description

2663 This function obtains supplementary service information, specified by “`reg_no`”, from non-volatile  
2664 memory.

2665

2666 See “Register supplementary service settings”.

2667 **38.Delete Service Information**

2668 **38.1 Symbol: celf\_mp\_cs\_del\_service\_info**

2669 **38.1.1 Syntax**

2670 CelfMpStatus celf\_mp\_cs\_del\_service\_info (  
2671 CelfMpRegNum reg\_no);

2672 **38.1.2 Argument**

2673 Name: reg\_no

2674 Type: CelfMpRegNum

2675 I/O: I

2676 Description:

2677 Registration number: 1 to 10

2678

2679 **38.1.3 Return Value**

2680 Type: CelfMpStatus

2681 Description:

2682 celf\_mp\_cs\_del\_service\_info() **shall** return one of the values defined:

2683 CELF\_MP\_STATUS\_OK: successful completion

2684 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2685

2686 **38.1.4 Include File**

2687 /usr/include/celf/mp\_cs.h

2688

2689 **38.1.5 Functional Description**

2690 This function deletes the supplementary service information specified by “reg\_no” from non-volatile  
2691 memory.

2692 **39.Remove Service Information**

2693 **39.1 Symbol: celf\_mp\_cs\_remove\_all\_service\_info**

2694 **39.1.1 Syntax**

2695 CelfMpStatus celf\_mp\_cs\_remove\_all\_service\_info (  
2696 void);

2697 **39.1.2 Argument**

2698 None.

2699

2700 **39.1.3 Return Value**

2701 Type: CelfMpStatus

2702 Description:

2703 celf\_mp\_cs\_remove\_all\_service\_info() **shall** return one of the values defined:

2704 CELF\_MP\_STATUS\_OK: successful completion

2705 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2706

2707 **39.1.4 Include File**

2708 /usr/include/celf/mp\_cs.h

2709

2710 **39.1.5 Functional Description**

2711 This function deletes all the supplementary service information from non-volatile memory.

## 2712 40.Set Response Message Settings

### 2713 40.1 Symbol: `celf_mp_cs_set_resp_msg`

#### 2714 40.1.1 Syntax

```
2715 CelfMpStatus celf_mp_cs_set_resp_msg (  
2716     CelfMpRegNum reg_no,  
2717     CelfMpCsSrvData * data);
```

#### 2718 40.1.2 Argument

2719 Name: `reg_no`

2720 Type: `CelfMpRegNum`

2721 I/O: `I`

2722 Description:

2723 Registration number: 1 to 10

2724 Name: `data`

2725 Type: `CelfMpCsSrvData`

2726 I/O: `I`

2727 Description:

2728 Pointer to the additional response message setting data area

2729

#### 2730 40.1.3 Return Value

2731 Type: `CelfMpStatus`

2732 Description:

2733 `celf_mp_cs_set_resp_msg()` **shall** return one of the values defined:

2734 `CELF_MP_STATUS_OK`: successful completion

2735 `CELF_MP_STATUS_ERR`: Other unsuccessful completion.

2736

#### 2737 40.1.4 Include File

2738 `/usr/include/celf/mp_cs.h`

2739

#### 2740 40.1.5 Functional Description

2741 This function registers the supplementary response message information for the supplementary service to  
2742 the non-volatile memory,

2743

2744 When a supplementary service is activated, and corresponding message from the network is received, this  
2745 supplementary response message is sent to the network.

2746



2747 The supplementary response message information is the service name and Dial data, which is response  
2748 message to send the network.

2749

2750 The dial data should be USSD.

2751

2752 The “reg\_no” is used as the key for accessing this supplementary response message .

2753 The value range is from 0 to 10.

2754

2755 For information about the structures, see section 0.1.

2756

**DRAFT**

## 2757 41. Get Response Message Settings

### 2758 41.1 Symbol: `celf_mp_cs_get_resp_msg`

#### 2759 41.1.1 Syntax

```
2760 CelfMpStatus celf_mp_cs_get_resp_msg (  
2761     CelfMpRegNum reg_no,  
2762     CelfMpCsSrvData * data);
```

#### 2763 41.1.2 Argument

2764 Name: `reg_no`

2765 Type: `CelfMpRegNum`

2766 I/O: `I`

2767 Description:

2768 Registration number: 1 to 10

2769 Name: `data`

2770 Type: `CelfMpCsSrvData`

2771 I/O: `I`

2772 Description:

2773 Pointer to the additional response message setting data area

2774

#### 2775 41.1.3 Return Value

2776 Type: `CelfMpStatus`

2777 Description:

2778 `celf_mp_cs_get_resp_msg()` shall return one of the values defined:

2779 `CELF_MP_STATUS_OK:` successful completion

2780 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2781

#### 2782 41.1.4 Include File

2783 `/usr/include/celf/mp_cs.h`

2784

#### 2785 41.1.5 Functional Description

2786 This function obtains the supplementary response message information, specified by “`reg_no`”, from non-  
2787 volatile memory.

2788

2789 See “Register response message settings”.

2790 **42.Delete Response Message Settings**

2791 **42.1 Symbol: celf\_mp\_cs\_del\_resp\_msg**

2792 **42.1.1 Syntax**

2793 CelfMpStatus celf\_mp\_cs\_del\_resp\_msg (  
2794 CelfMpRegNum reg\_no);

2795 **42.1.2 Argument**

2796 Name: reg\_no

2797 Type: CelfMpRegNum

2798 I/O: I

2799 Description:

2800 Registration number: 1 to 10

2801

2802 **42.1.3 Return Value**

2803 Type: CelfMpStatus

2804 Description:

2805 celf\_mp\_cs\_del\_resp\_msg() **shall** return one of the values defined:

2806 CELF\_MP\_STATUS\_OK: successful completion

2807 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2808

2809 **42.1.4 Include File**

2810 /usr/include/celf/mp\_cs.h

2811

2812 **42.1.5 Functional Description**

2813 This function deletes the supplementary response message information, specified by “reg\_no”, from non-  
2814 volatile memory.

## 2815 **43.Remove All Response Message Settings**

### 2816 **43.1 Symbol: celf\_mp\_cs\_remove\_all\_resp\_msg**

#### 2817 **43.1.1 Syntax**

2818 CelfMpStatus celf\_mp\_cs\_remove\_all\_resp\_msg (  
2819 void);

#### 2820 **43.1.2 Argument**

2821 None.

2822

#### 2823 **43.1.3 Return Value**

2824 Type: CelfMpStatus

2825 Description:

2826 celf\_mp\_cs\_remove\_all\_resp\_msg() **shall** return one of the values defined:

2827 CELF\_MP\_STATUS\_OK: successful completion

2828 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2829

#### 2830 **43.1.4 Include File**

2831 /usr/include/celf/mp\_cs.h

2832

#### 2833 **43.1.5 Functional Description**

2834 This function removes from non-volatile memory all the supplementary response message information.

2835

## 2836 44.Set Reconnection Tone

### 2837 44.1 Symbol: `celf_mp_cs_set_reconnection_tone`

#### 2838 44.1.1 Syntax

```
2839 CelfMpStatus celf_mp_cs_set_reconnection_tone (  
2840     CelfMpCsReconnectionTone     reconn);
```

#### 2841 44.1.2 Argument

2842 Name: `reconn`

2843 Type: `CelfMpCsReconnectionTone`

2844 I/O: `I`

2845 Description:

2846 Reconnection tone to be set

2847 `CELF_MP_CS_RECONN_ON_T_OFF:` Tone OFF

2848 `CELF_MP_CS_RECONN_ON_T_LOW:` Tone ON low tone

2849 `CELF_MP_CS_RECONN_ON_T_HI:` Tone ON high tone

2850

#### 2851 44.1.3 Return Value

2852 Type: `CelfMpStatus`

2853 Description:

2854 `celf_mp_cs_set_reconnection_tone()` shall return one of the values defined:

2855 `CELF_MP_STATUS_OK:` successful completion

2856 `CELF_MP_STATUS_ERR:` Other unsuccessful completion.

2857

#### 2858 44.1.4 Include File

2859 `/usr/include/celf/mp_cs.h`

2860

#### 2861 44.1.5 Functional Description

2862 This function sets the reconnection tone information to the non-volatile memory.

2863

2864 **45.Get Reconnection Tone**

2865 **45.1 Symbol: celf\_mp\_cs\_get\_reconnection\_tone**

2866 **45.1.1 Syntax**

2867 CelfMpCsReconnectionTone celf\_mp\_cs\_get\_reconnection\_tone (  
2868 void);

2869 **45.1.2 Argument**

2870 None.

2871

2872 **45.1.3 Return Value**

2873 Type: CelfMpCsReconnectionTone

2874 I/O: O

2875 Description:

2876 celf\_mp\_cs\_get\_reconnection\_tone() **shall** return one of the values defined:

2877 CELF\_MP\_CS\_RECONN\_ON\_T\_OFF: Tone OFF

2878 CELF\_MP\_CS\_RECONN\_ON\_T\_LOW: Tone ON low tone

2879 CELF\_MP\_CS\_RECONN\_ON\_T\_HI: Tone ON high tone

2880

2881

2882 **45.1.4 Include File**

2883 /usr/include/celf/mp\_cs.h

2884

2885 **45.1.5 Functional Description**

2886 This function gets the reconnection tone information to the non-volatile memory.

2887 **46.Get Noise Cancel**

2888 **46.1 Symbol: celf\_mp\_cs\_get\_noise\_cancel**

2889 **46.1.1 Syntax**

2890 CelfMpCsNoiseCancel celf\_mp\_cs\_get\_noise\_cancel (  
2891 void);

2892 **46.1.2 Argument**

2893 None.

2894

2895 **46.1.3 Return Value**

2896 Type: CelfMpCsNoiseCancel

2897 I/O: O

2898 Description:

2899 `celf_mp_cs_get_noise_cancel()` **shall** return one of the values defined:

2900 CELF\_MP\_CS\_ON: Noise canceller ON

2901 CELF\_MP\_CS\_OFF: Noise canceller OFF

2902

2903 **46.1.4 Include File**

2904 `/usr/include/celf/mp_cs.h`

2905

2906 **46.1.5 Functional Description**

2907 This function gets the noise canceller status.

2908

2909 **47.Set Noise Cancel**

2910 **47.1 Symbol: celf\_mp\_cs\_set\_noise\_cancel**

2911 **47.1.1 Syntax**

2912 CelfMpStatus celf\_mp\_cs\_set\_noise\_cancel (  
2913 CelfMpCsNoiseCancel mode);

2914 **47.1.2 Argument**

2915 Name: mode

2916 Type: CelfMpCsNoiseCancel

2917 I/O: I

2918 Description:

2919 Reconnection tone to be set

2920 CELF\_MP\_CS\_ON: Noise canceller ON

2921 CELF\_MP\_CS\_OFF: Noise canceller OFF

2922

2923 **47.1.3 Return Value**

2924 Type: CelfMpStatus

2925 Description:

2926 `celf_mp_cs_set_noise_cancel()` shall return one of the values defined:

2927 CELF\_MP\_STATUS\_OK: successful completion

2928 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2929

2930 **47.1.4 Include File**

2931 `/usr/include/celf/mp_cs.h`

2932

2933 **47.1.5 Functional Description**

2934 This function sets the noise canceller off or on.



2935 **48.Get Quality Alarm**

2936 **48.1 Symbol: celf\_mp\_cs\_get\_quality\_alarm**

2937 **48.1.1 Syntax**

2938 CelfMpCsQualAlarm celf\_mp\_cs\_get\_quality\_alarm(  
2939 void);

2940 **48.1.2 Argument**

2941 None.

2942

2943 **48.1.3 Return Value**

2944 Type: CelfMpCsQualAlarm

2945 I/O: O

2946 Description:

2947 celf\_mp\_cs\_get\_quality\_alarm() **shall** return one of the values defined:

2948 CELF\_MP\_CS\_QUALITY\_ALM\_OFF: Quality alarm OFF

2949 CELF\_MP\_CS\_QUALITY\_ALM\_LOW: Quality alarm ON low tone

2950 CELF\_MP\_CS\_QUALITY\_ALM\_HI: Quality alarm ON high tone

2951

2952 **48.1.4 Include File**

2953 /usr/include/celf/mp\_cs.h

2954

2955 **48.1.5 Functional Description**

2956 This function gets the status of the call quality alarm sound.

2957 **49.Set Quality Alarm**

2958 **49.1 Symbol: celf\_mp\_cs\_set\_quality\_alarm**

2959 **49.1.1 Syntax**

2960 CelfMpStatus celf\_mp\_cs\_set\_quality\_alarm (  
2961 CelfMpCsQualAlarm mode);

2962 **49.1.2 Argument**

2963 Name: mode

2964 Type: CelfMpCsQualAlarm

2965 I/O: I

2966 Description:

2967 CELF\_MP\_CS\_QUALITY\_ALM\_OFF: Quality alarm OFF

2968 CELF\_MP\_CS\_QUALITY\_ALM\_LOW: Quality alarm ON low tone

2969 CELF\_MP\_CS\_QUALITY\_ALM\_HI: Quality alarm ON high tone

2970

2971 **49.1.3 Return Value**

2972 Type: CelfMpStatus

2973 Description:

2974 celf\_mp\_cs\_set\_quality\_alarm() **shall** return one of the values defined:

2975 CELF\_MP\_STATUS\_OK: successful completion

2976 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

2977

2978 **49.1.4 Include File**

2979 /usr/include/celf/mp\_cs.h

2980

2981 **49.1.5 Functional Description**

2982 This function sets the call quality alarm sound.

2983 **50.Get Noise Cancel Permit**

2984 **50.1 Symbol: celf\_mp\_cs\_get\_noise\_cancel\_permit**

2985 **50.1.1 Syntax**

2986 CelfMpCsNoiseCancel celf\_mp\_cs\_get\_noise\_cancel\_permit(  
2987 void);

2988 **50.1.2 Argument**

2989 None.

2990

2991 **50.1.3 Return Value**

2992 Type: CelfMpCsNoiseCancel

2993 I/O: O

2994 Description:

2995 `celf_mp_cs_get_noise_cancel_permit()` shall return one of the values defined:

2996 CELF\_MP\_CS\_ON: Noise canceller permission

2997 CELF\_MP\_CS\_OFF: Noise canceller non-permission

2998

2999 **50.1.4 Include File**

3000 `/usr/include/celf/mp_cs.h`

3001

3002 **50.1.5 Functional Description**

3003 This function obtains whether noise canceller is permitted or not.

3004 **51.Set High Priority communication mode**

3005 **51.1 Symbol: celf\_mp\_cs\_set\_hi\_prio\_com**

3006 **51.1.1 Syntax**

3007 CelfMpStatus celf\_mp\_cs\_set\_hi\_prio\_com (  
3008 CelfMpCsHiPrioCom mode);

3009 **51.1.2 Argument**

3010 Name: mode

3011 Type: CelfMpCsHiPrioCom

3012 I/O: I

3013 Description:

3014 Reconnection tone to be set

3015 CELF\_MP\_CS\_COMPRI\_NONE: No setting

3016 CELF\_MP\_CS\_COMPRI\_VOICE: Voice

3017 CELF\_MP\_CS\_COMPRI\_PACKET: Packet

3018

3019 **51.1.3 Return Value**

3020 Type: CelfMpStatus

3021 Description:

3022 celf\_mp\_cs\_set\_hi\_prio\_com() **shall** return one of the values defined:

3023 CELF\_MP\_STATUS\_OK: successful completion

3024 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3025

3026 **51.1.4 Include File**

3027 /usr/include/celf/mp\_cs.h

3028

3029 **51.1.5 Functional Description**

3030 This function sets the high priority communication mode either on the voice communication or on the  
3031 packet communication.

## 3032 **52. Get Phone Answering Sound Activation**

### 3033 **52.1 Symbol: celf\_mp\_cs\_get\_vm\_sound\_status**

#### 3034 **52.1.1 Syntax**

3035 CelfMpCsVmSound celf\_mp\_cs\_get\_vm\_sound\_status(  
3036 void);

#### 3037 **52.1.2 Argument**

3038 None.

3039

#### 3040 **52.1.3 Return Value**

3041 Type: CelfMpCsVmSound

3042 I/O: O

3043 Description:

3044 `celf_mp_cs_get_vm_sound_status()` **shall** return one of the values defined:

3045 CELF\_MP\_CS\_ON: Message sound ON

3046 CELF\_MP\_CS\_OFF: Message sound OFF

3047

#### 3048 **52.1.4 Include File**

3049 `/usr/include/celf/mp_cs.h`

3050

#### 3051 **52.1.5 Functional Description**

3052 This function gets the setting status.

3053 IF the setting status is ON, the phone sounds, when the number of voice mail system is increased.

## 3054 **53.Set Phone Answering Sound Activation**

### 3055 **53.1 Symbol: celf\_mp\_cs\_set\_vm\_sound\_status**

#### 3056 **53.1.1 Syntax**

3057 CelfMpStatus celf\_mp\_cs\_get\_vm\_sound\_status (  
3058 CelfMpCsVmSound mode);

#### 3059 **53.1.2 Argument**

3060 Type: CelfMpCsVmSound

3061 I/O: O

3062 Description:

3063 CELF\_MP\_CS\_ON: Message sound ON

3064 CELF\_MP\_CS\_OFF: Message sound OFF

3065

#### 3066 **53.1.3 Return Value**

3067 Type: CelfMpStatus

3068 Description:

3069 celf\_mp\_cs\_set\_vm\_sound\_status( ) **shall** return one of the values defined:

3070 CELF\_MP\_STATUS\_OK: successful completion

3071 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3072

#### 3073 **53.1.4 Include File**

3074 /usr/include/celf/mp\_cs.h

3075

#### 3076 **53.1.5 Functional Description**

3077 This function sets the phone sounds status whether the phone sounds or not.

3078 **54.Get Automatic Receive Status**

3079 **54.1 Symbol: celf\_mp\_cs\_get\_auto\_rcv\_status**

3080 **54.1.1 Syntax**

3081 CelfMpCsVmSound celf\_mp\_cs\_get\_auto\_rcv\_status(  
3082 void);

3083 **54.1.2 Argument**

3084 None.

3085

3086 **54.1.3 Return Value**

3087 Type: CelfMpCsVmSound

3088 I/O: O

3089 Description:

3090 celf\_mp\_cs\_get\_auto\_rcv\_status() **shall** return one of the values defined:

3091 CELF\_MP\_CS\_ON: Automatic incoming call ON

3092 CELF\_MP\_CS\_OFF: Automatic incoming call OFF

3093

3094 **54.1.4 Include File**

3095 /usr/include/celf/mp\_cs.h

3096

3097 **54.1.5 Functional Description**

3098 This function obtains the status of automatic incoming call.

3099 The status is ON or OFF.

## 3100 **55.Set Automatic Receive Status**

### 3101 **55.1 Symbol: celf\_mp\_cs\_set\_auto\_rcv\_status**

#### 3102 **55.1.1 Syntax**

3103 CelfMpStatus celf\_mp\_cs\_set\_auto\_rcv\_status (  
3104 CelfMpCsAutoRcv mode);

#### 3105 **55.1.2 Argument**

3106 Type: CelfMpCsAutoRcv

3107 I/O: O

3108 Description:

3109 CELF\_MP\_CS\_ON: Automatic incoming call ON

3110 CELF\_MP\_CS\_OFF: Automatic incoming call OFF

3111

#### 3112 **55.1.3 Return Value**

3113 Type: CelfMpStatus

3114 Description:

3115 celf\_mp\_cs\_set\_auto\_rcv\_status( ) **shall** return one of the values defined:

3116 CELF\_MP\_STATUS\_OK: successful completion

3117 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3118

#### 3119 **55.1.4 Include File**

3120 /usr/include/celf/mp\_cs.h

3121

#### 3122 **55.1.5 Functional Description**

3123 This function sets the automatic incoming call status.



3124 **56.Get Automatic Timer**

3125 **56.1 Symbol: celf\_mp\_cs\_get\_auto\_timer**

3126 **56.1.1 Syntax**

3127 CelfMpCsTimer celf\_mp\_cs\_get\_auto\_timer(  
3128 void);

3129 **56.1.2 Argument**

3130 None.

3131

3132 **56.1.3 Return Value**

3133 Type: CelfMpCsTimer

3134 I/O: O

3135 Description:

3136 celf\_mp\_cs\_get\_auto\_timer() **shall** return one of the values defined:

3137 1 to 120 seconds

3138

3139 **56.1.4 Include File**

3140 /usr/include/celf/mp\_cs.h

3141

3142 **56.1.5 Functional Description**

3143 This function obtains the timer value of the automatic incoming call.

3144 The timer value is the duration of sounding of the ring alert.

3145 **57.Set Automatic Timer**

3146 **57.1 Symbol: celf\_mp\_cs\_set\_auto\_timer**

3147 **57.1.1 Syntax**

3148 CelfMpStatus celf\_mp\_cs\_set\_auto\_timer (  
3149 CelfMpCsTimer time);

3150 **57.1.2 Argument**

3151 Type: CelfMpCsTimer

3152 I/O: O

3153 Description:

3154 1 to 120 seconds

3155

3156 **57.1.3 Return Value**

3157 Type: CelfMpStatus

3158 Description:

3159 celf\_mp\_cs\_set\_auto\_timer() shall return one of the values defined:

3160 CELF\_MP\_STATUS\_OK: successful completion

3161 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3162

3163 **57.1.4 Include File**

3164 /usr/include/celf/mp\_cs.h

3165

3166 **57.1.5 Functional Description**

3167 This function sets the timer value of the automatic incoming call.

3168

3169

3170

3171

3172 **58.Get Reset Date**

3173 **58.1 Symbol: celf\_mp\_cs\_get\_reset\_date**

3174 **58.1.1 Syntax**

3175 CelfMpStatus celf\_mp\_cs\_get\_reset\_date(  
3176 CelfMpCsDate \* reset\_date);

3177 **58.1.2 Argument**

3178 Type: CelfMpCsDate

3179 I/O: O

3180 Description:

3181 Accumulated date record

3182 See section 0.1 for details.

3183

3184

3185 **58.1.3 Return Value**

3186 Type: CelfMpStatus

3187 Description:

3188 `celf_mp_cs_get_reset_date()` shall return one of the values defined:

3189 CELF\_MP\_STATUS\_OK: successful completion

3190 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3191

3192 **58.1.4 Include File**

3193 `/usr/include/celf/mp_cs.h`

3194

3195 **58.1.5 Functional Description**

3196 This function obtains the date and time when the accumulated call duration was reset.

3197 The value is obtained from non-volatile memory.

3198 **59.Set Reset Date**

3199 **59.1 Symbol: celf\_mp\_cs\_set\_reset\_date**

3200 **59.1.1 Syntax**

3201 CelfMpStatus celf\_mp\_cs\_set\_reset\_date(  
3202 void);

3203 **59.1.2 Argument**

3204 None.

3205

3206 **59.1.3 Return Value**

3207 Type: CelfMpStatus

3208 Description:

3209 celf\_mp\_cs\_set\_reset\_date() **shall** return one of the values defined:

3210 CELF\_MP\_STATUS\_OK: successful completion

3211 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3212

3213 **59.1.4 Include File**

3214 /usr/include/celf/mp\_cs.h

3215

3216 **59.1.5 Functional Description**

3217 This function sets the current date and time as the reset date and time of the accumulated call duration.

3218 The value set to non-volatile memory.

## 3219 **60.Get Call Silent Time**

### 3220 **60.1 Symbol: celf\_mp\_cs\_get\_call\_silent\_time**

#### 3221 **60.1.1 Syntax**

3222 CelfMpTime celf\_mp\_cs\_get\_call\_silent\_time(  
3223 void );

#### 3224 **60.1.2 Argument**

3225 None.

3226

3227

#### 3228 **60.1.3 Return Value**

3229 Type: CelfMpTime

3230 I/O: O

3231 Description:

3232 `celf_mp_cs_get_call_silent_time()` **shall** return one of the values defined:

3233 0 to 99 seconds

3234

#### 3235 **60.1.4 Include File**

3236 `/usr/include/celf/mp_cs.h`

3237

#### 3238 **60.1.5 Functional Description**

3239 This function gets the duration between the arrival of incoming call and the start of sounding of the ring  
3240 alert. This duration is called the silent time.

3241 This function is effective that the number of this incoming call is unregistered with the phone book.

3242 **61.Set Call Silent Time**

3243 **61.1 Symbol: celf\_mp\_cs\_set\_call\_silent\_time**

3244 **61.1.1 Syntax**

3245 CelfMpStatus celf\_mp\_cs\_set\_call\_silent\_time(  
3246 CelfMpCsTimer time);

3247 **61.1.2 Argument**

3248 Type: CelfMpCsTimer

3249 I/O: I

3250 Description:

3251 0 to 99 seconds

3252

3253 **61.1.3 Return Value**

3254 Type: CelfMpStatus

3255 Description:

3256 celf\_mp\_cs\_set\_call\_silent\_time() **shall** return one of the values defined:

3257 CELF\_MP\_STATUS\_OK: successful completion

3258 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3259

3260 **61.1.4 Include File**

3261 /usr/include/celf/mp\_cs.h

3262

3263 **61.1.5 Functional Description**

3264 This function sets the silent time.

3265 Refer to get calling operation start time.

3266 **62. Get Call Recorded**

3267 **62.1 Symbol: celf\_mp\_cs\_get\_call\_recorded**

3268 **62.1.1 Syntax**

3269 CelfMpSetting celf\_mp\_cs\_get\_call\_recorded(  
3270 void );

3271 **62.1.2 Argument**

3272 None.

3273

3274

3275 **62.1.3 Return Value**

3276 Type: CelfMpSetting

3277 I/O: O

3278 Description:

3279 celf\_mp\_cs\_get\_call\_recorded() **shall** return one of the values defined:

3280 CELF\_MP\_CS\_ON: Setting ON

3281 CELF\_MP\_CS\_OFF: Setting OFF

3282

3283 **62.1.4 Include File**

3284 /usr/include/celf/mp\_cs.h

3285

3286 **62.1.5 Functional Description**

3287 This function gets the setting condition of whether the silent call is recorded in the absent incoming call  
3288 log, or not.

3289 The absent incoming call log is the log that records no-responded incoming call.

3290 The silent call is the incoming call, which disconnects within the silent time.

3291 Refer to “Get calling operation start time”.

3292

3293

3294

3295 **63.Set Call Recorded**

3296 **63.1 Symbol: celf\_mp\_cs\_set\_call\_recorded**

3297 **63.1.1 Syntax**

3298 CelfMpStatus celf\_mp\_cs\_set\_call\_recorded(  
3299 CelfMpCsSetting mode);

3300 **63.1.2 Argument**

3301 Type: CelfMpCsSetting

3302 I/O: O

3303 Description:

3304 CELF\_MP\_CS\_ON: Setting ON

3305 CELF\_MP\_CS\_OFF: Setting OFF

3306

3307 **63.1.3 Return Value**

3308 Type: CelfMpStatus

3309 Description:

3310 celf\_mp\_cs\_set\_call\_start\_time() **shall** return one of the values defined:

3311 CELF\_MP\_STATUS\_OK: successful completion

3312 CELF\_MP\_STATUS\_ERR: Other unsuccessful completion.

3313

3314 **63.1.4 Include File**

3315 /usr/include/celf/mp\_cs.h

3316

3317 **63.1.5 Functional Description**

3318 This function sets the setting condition of whether the silent call is recorded in the absent incoming call log,  
3319 or not.

3320 Refer to “Get recording condition to absent incoming call log”.

3321