DSHEng5 装置／ホスト通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHEng5 GEM 通信エンジン・クラス説明書

Vol－4

SECS-II メッセージ送信クラス

２０１９年１２月（改訂-1）

株式会社データマップ

文書番号　DSHENG5-19-30324-01

［取り扱い注意］

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、（株）データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2019-06-28	初版	
2.	2019.12.16	訂正と記述追加	誤字等の訂正 説明の追加など。仕様上は変更はない。
3.			
4.			

目　　次

１．はじめに.....1
［　SECS-Ⅱメッセージ一覧表　］.....2
２．SECS-II メッセージ送受信の仕方.....5
２．１　１次メッセージの送信と２次メッセージ受信.....5
２．１．１　非ブロックモードのプログラミング例.....6
２．１．２　ブロックモードのプログラミング.....8
２．２　受信１次メッセージに対する２次メッセージの応答の仕方.....9
３．装置定数(EC)関連メッセージの送受信.....11
３．１　class_SendS2F13 クラス - Equipment Constant Request.....12
３．１．１　コンストラクタ.....12
３．１．２　プロパティ.....12
３．１．３　メソッド.....12
３．１．３．１　SendS2F13() - 非ブロックモードでの送信.....13
３．１．３．２　SendS2F13_wait() - ブロックモードでの送信.....14
３．２　class_SendS2F15 クラス - New Equipment Constant Send.....15
３．２．１　コンストラクタ.....15
３．２．２　プロパティ.....15
３．２．３　メソッド.....15
３．２．３．１　SendS2F15().....16
３．３．４．２　SendS2F15_wait().....17
３．３　class_SendS2F29 クラス - Equipmeny Constant Namelist Request.....18
３．３．１　コンストラクタ.....18
３．３．２　プロパティ.....18
３．３．３　メソッド.....18
３．３．３．１　SendS2F29().....19
３．３．３．２　SendS2F29_wait().....20
４．装置状態変数(SV)関連メッセージの送受信.....21
４．１　class_SendS1F3 クラス - Selected Equipment Status Request.....22
４．１．１　コンストラクタ.....22
４．１．２　プロパティ.....22
４．１．３　メソッド.....22
４．１．３．１　SendS1F3().....23
４．１．３．２　SendS1F3_wait().....24
４．２　class_SendS1F11 クラス - Status Variable Namelist Request.....25
４．２．１　コンストラクタ.....25
４．２．２　プロパティ.....25
４．２．３　メソッド.....25
４．２．３．１　SendS1F11().....26
４．２．３．２　SendS1F11_wait().....27
５．変数リミット(LIMIT)関連メッセージの送受信.....28
５．１　class_SendS2F45 クラス - Define Variable Limit Attributes.....29
５．１．１　コンストラクタ.....29
５．１．２　プロパティ.....29
５．１．３　メソッド.....29
５．１．３．１　SendS2F45().....30
５．１．３．２　SendS2F45_wait().....31
５．２　class_SendS2F47 クラス - Variable Limit Attributes Request.....32

５．２．１　コンストラクタ 32
５．２．２　プロパティ 32
５．２．３　メソッド 32
５．２．３．１　SendS2F47() 33
５．２．３．２　SendS2F47_wait() 34
６．トレース(TRACE)関連メッセージの送受信 35
６．１　class_SendS2F23 クラス - Trace Initialize Send 36
６．１．１　コンストラクタ 36
６．１．２　プロパティ 36
６．１．３　メソッド 36
６．１．３．１　SendS2F23() 37
６．１．３．２　SendS2F23_wait() 38
７．収集イベントとレポート関連メッセージの送受信 39
７．１　class_SendS2F35 クラス - Link Event Report 40
７．１．１　コンストラクタ 40
７．１．２　プロパティ 40
７．１．３　メソッド 40
７．１．３．１　SendS2F35() 41
７．１．３．２　SendS2F35_wait() 42
７．２　class_SendS2F37 クラス - Enable/Disabel Event Report 43
７．２．１　コンストラクタ 43
７．２．２　プロパティ 43
７．２．３　メソッド 43
７．２．３．１　SendS2F37() 44
７．２．３．２　SendS2F37_wait() 45
７．３　class_SendS2F33 クラス - Define Report 46
７．３．１　コンストラクタ 46
７．３．２　プロパティ 46
７．３．３　メソッド 46
７．３．３．１　SendS2F33() 47
７．３．３．２　SendS2F33_wait() 48
７．４　class_SendS6F11 クラス - Event Report Send 49
７．４．１　コンストラクタ 49
７．４．２　プロパティ 49
７．４．３　メソッド 49
７．４．３．１　SendS6F11() 50
７．４．３．２　SendS6F11_wait() 51
７．５　class_SendS6F15 クラス - Event Report Request 52
７．５．１　コンストラクタ 52
７．５．２　プロパティ 52
７．５．３　メソッド 52
７．５．３．１　SendS6F15() 53
７．５．３．２　SendS6F15_wait() 54
７．６　class_SendS6F19 クラス - Indivisual Report Data 55
７．６．１　コンストラクタ 55
７．６．２　プロパティ 55
７．６．３　メソッド 55
７．６．３．１　SendS6F19() 56
７．６．３．２　SendS6F19_wait() 57

8．アラーム関連メッセージの送受信 58
8．1　class_SendS5F1 クラス - Alarm Report Send 58
8．1．1　コンストラクタ 58
8．1．2　プロパティ 58
8．1．3　メソッド 58
8．1．3．1　SendS5F1() 59
8．1．3．2　SendS5F1_wait() 60
8．2　class_SendS5F3 クラス - Enable/Disabel Alarm Send 61
8．2．1　コンストラクタ 61
8．2．2　プロパティ 61
8．2．3　メソッド 61
8．2．3．1　SendS5F3() 62
8．2．3．2　SendS5F3_wait() 63
8．3　class_SendS5F5 クラス - List Alarm Reques 64
8．3．1　コンストラクタ 64
8．3．2　プロパティ 64
8．3．3　メソッド 64
8．3．3．1　SendS5F5() 65
8．3．3．2　SendS5F5_wait() 66
9．プロセスプログラム関連メッセージの送受信 67
9．1　class_SendS7F1 クラス - Process Program Load Inquire 68
9．1．1　コンストラクタ 68
9．1．2　プロパティ 68
9．1．3　メソッド 68
9．1．3．1　SendS7F1() 69
9．1．3．2　SendS7F1_wait() 70
9．2　class_SendS7F3 クラス - Process Program Send 71
9．2．1　コンストラクタ 71
9．2．2　プロパティ 71
9．2．3　メソッド 71
9．2．3．1　SendS7F3() 72
9．2．3．2　SendS7F3_wait() 73
9．3　class_SendS7F5 クラス - Process Program Data 74
9．3．1　コンストラクタ 74
9．3．2　プロパティ 74
9．3．3　メソッド 74
9．3．3．1　SendS7F5() 75
9．3．3．2　SendS7F5_wait() 76
9．4　class_SendS7F17 クラス - Delete Process Program Send 77
9．4．1　コンストラクタ 77
9．4．2　プロパティ 77
9．4．3　メソッド 77
9．4．3．1　SendS7F17() 78
9．4．3．2　SendS1F17_wait() 79
9．5　class_SendS7F19 クラス - Current EPPD Request 80
9．5．1　コンストラクタ 80
9．5．2　プロパティ 80
9．5．3　メソッド 80
9．5．3．1　SendS7F19() 81

9．5．3．2　SendS7F19_wait() 82
10. フォーマット付きプロセスプログラム関連メッセージの送受信 83
10. 1　class_SendS7F23 クラス-Formatted Process Program Send 84
10. 1．1　コンストラクタ 84
10. 1．2　プロパティ 84
10. 1．3　メソッド 84
10. 1．3．1　SendS7F23() 85
10. 1．3．2　SendS7F23_wait() 86
10. 2　class_SendS7F25 クラス -Formatted Process Program Data 87
10. 2．1　コンストラクタ 87
10. 2．2　プロパティ 87
10. 2．3　メソッド 87
10. 2．3．1　SendS7F25() 88
10. 2．3．2　SendS7F25_wait() 89
11. レシピ関連メッセージの送受信 90
11. 1　class_SendS15F3 クラス - Recipe Name Space Action Request 91
11. 1．1　コンストラクタ 91
11. 1．2　プロパティ 91
11. 1．3　メソッド 91
11. 1．3．1　SendS15F3() 92
11. 1．3．2　SendS15F3_wait() 93
11. 2　class_SendS15F5 クラス -Recipe Name space Rename Request 94
11. 2．1　コンストラクタ 94
11. 2．2　プロパティ 94
11. 2．3　メソッド 94
11. 2．3．2　SendS15F5() 95
11. 2．3．3　Send_S15F5_wait() 96
11. 3　class_SendS15F7 クラス 97
11. 3．1　コンストラクタ 97
11. 3．2　プロパティ 97
11. 3．3　メソッド 97
11. 3．3．1　SendS15F7() 98
11. 3．3．2　SendS15F7_wait() 99
11. 4　class_SendS15F9 クラス -Recipe Status Request 100
11. 4．1　コンストラクタ 100
11. 4．2　プロパティ 100
11. 4．3　メソッド 100
11. 4．3．1　SendS15F9() 101
11. 4．3．2　SendS15F9_wait() 102
11. 5　class_SendS15F13 クラス -Recipe Create Request 103
11. 5．1　コンストラクタ 103
11. 5．2　プロパティ 103
11. 5．3　メソッド 103
11. 5．3．1　SendS15F13() 104
11. 5．3．2　SendS15F13_wait() 105
11. 6　class_SendS15F17 クラス -Recipe Retrieve Request 106
11. 6．1　コンストラクタ 106
11. 6．2　プロパティ 106
11. 6．3　メソッド 106

11. 6. 3. 1 SendS15F17() 107
11. 6. 3. 2 SendS15F17_wait() 108
12. プロセス・ジョブ関連メッセージの送受信 109
12. 1 class_SendS16F5 クラス -Process Job Command Request 110
12. 1. 1 コンストラクタ 110
12. 1. 2 プロパティ 110
12. 1. 3 メソッド 110
12. 1. 3. 1 SendS16F50 111
12. 1. 3. 2 SendS16F5_wait() 112
12. 2 class_SendS16F11 クラス -PrJobCreateEnh 113
12. 2. 1 コンストラクタ 113
12. 2. 2 プロパティ 113
12. 2. 3 メソッド 113
12. 2. 3. 1 SendS16F11() 114
12. 2. 3. 2 SendS16F11_wait() 116
12. 3 class_SendS16F15 クラス -PrJobMultiCreate 117
12. 3. 1 コンストラクタ 117
12. 3. 2 プロパティ 117
12. 3. 3 メソッド 117
12. 3. 3. 1 SendS16F15() 118
12. 3. 3. 2 Send16F15_wait() 119
12. 4 class_SendS16F17 クラス -PrJobDeque 120
12. 4. 1 コンストラクタ 120
12. 4. 2 プロパティ 120
12. 4. 3 メソッド 120
12. 4. 3. 1 SendS16F17() 121
12. 4. 3. 2 SendS16F17_wait() 122
12. 5 class_SendS16F19 クラス -PrGetAllJobs 123
12. 5. 1 コンストラクタ 123
12. 5. 2 プロパティ 123
12. 5. 3 メソッド 123
12. 5. 3. 1 SendS16F19() 124
12. 5. 3. 2 SendS16F19_wait() 125
12. 6 class_SendS16F21 クラス -PrGetSpace 126
12. 6. 1 コンストラクタ 126
12. 6. 2 プロパティ 126
12. 6. 3 メソッド 126
12. 6. 3. 1 SendS16F21() 127
12. 6. 3. 2 SendS16F21_wait() 128
13. コントロール・ジョブ関連メッセージの送受信 129
13. 1 class_SendS14F9 クラス -Create Object Request 130
13. 1. 1 コンストラクタ 130
13. 1. 2 プロパティ 130
13. 1. 3 メソッド 130
13. 1. 3. 1 SendS14F9() - 131
13. 1. 3. 2 Send S14F9_wait() 132
13. 2 class_SendS14F11 クラス -Delete Object Request 133
13. 2. 1 コンストラクタ 133
13. 2. 2 プロパティ 133

13. 2. 3 メソッド 133
13. 2. 3. 1 SendS14F11() 134
13. 2. 3. 2 SendS14F11_wait() 135
13. 3 class_SendS16F27 クラス - Control Job Command Request 136
13. 3. 1 コンストラクタ 136
13. 3. 2 プロパティ 136
13. 3. 3 メソッド 136
13. 3. 3. 1 SendS16F27() 137
13. 3. 3. 2 SendS16F27_wait() 138
14. スプール関連メッセージの送受信 139
14. 1 class_SendS2F43 クラス - Reset Spooling Stream and Function 140
14. 1. 1 コンストラクタ 140
14. 1. 2 プロパティ 140
14. 1. 3 メソッド 140
14. 1. 3. 1 SendS2F43() 141
14. 1. 3. 2 SendS2F43_wait() 142
14. 2 class_SendS6F23 クラス - Request Spooled Data 143
14. 2. 1 コンストラクタ 143
14. 2. 2 プロパティ 143
14. 2. 3 メソッド 143
14. 2. 3. 1 SendS6F23() 144
14. 2. 3. 2 SendS6F23_wait() 145
15. ホストコマンド、キャリアアクション関連メッセージの送受信 146
15. 1 class_SendS2F41 クラス - Host Command Send 147
15. 1. 1 コンストラクタ 147
15. 1. 2 プロパティ 147
15. 1. 3 メソッド 147
15. 1. 3. 1 SendS2F41() 148
15. 1. 3. 2 SendS2F41_wait() 149
15. 2 class_SendS2F49 クラス - Enhanced Remote Command 150
15. 2. 1 コンストラクタ 150
15. 2. 2 プロパティ 150
15. 2. 3 メソッド 150
15. 2. 3. 1 SendS2F49() 151
15. 2. 3. 2 SendS2F49_wait() 152
15. 3 class_SendS3F17 クラス - Carrier Action Request 153
15. 3. 1 コンストラクタ 153
15. 3. 2 プロパティ 153
15. 3. 3 メソッド 153
15. 3. 3. 1 SendS3F17() 154
15. 3. 3. 2 SendS3F17_wait() 155
15. 4 class_SendS3F23 クラス - Port Group Action Request 156
15. 4. 1 コンストラクタ 156
15. 4. 2 プロパティ 156
15. 4. 3 メソッド 156
15. 4. 3. 1 SendS3F23() 157
15. 4. 3. 2 SendS3F23_wait() 158
15. 5 class_SendS3F25 クラス - Port Action Request 159
15. 5. 1 コンストラクタ 159

15. 5. 2 プロパティ 159
15. 5. 3 メソッド 159
15. 5. 3. 1 SendS3F25() 160
15. 5. 3. 2 SendS3F25_wait() 161
15. 6 class_SendS3F27 クラス - Change Access 162
15. 6. 1 コンストラクタ 162
15. 6. 2 プロパティ 162
15. 6. 3 メソッド 162
15. 6. 3. 1 SendS3F27() 163
15. 6. 3. 2 SendS3F27_wait() 164
16. 端末表示関連メッセージの送受信 165
16. 1 class_SendS10F1 クラス - S10F1 送信 Terminal Request 166
16. 1. 1 コンストラクタ 166
16. 1. 2 プロパティ 166
16. 1. 3 メソッド 166
16. 1. 3. 1 SendS10F1() 167
16. 1. 3. 2 SendS10F1_wait() 168
16. 2 class_SendS10F3 クラス - Treminal Display, Single 169
16. 2. 1 コンストラクタ 169
16. 2. 2 プロパティ 169
16. 2. 3 メソッド 169
16. 2. 3. 1 SendS10F3() 170
16. 2. 3. 2 SendS10F3_wait() 171
16. 3 class_SendS10F5 クラス - Terminal Display, Multi-Block 172
16. 3. 1 コンストラクタ 172
16. 3. 2 プロパティ 172
16. 3. 3 メソッド 172
16. 3. 3. 1 SendS10F5() 173
16. 3. 3. 2 SendS10F5_wait() 174
17. オンライン／オフライン、日付時刻設定メッセージ送受信 175
17. 1 class_SendS1F15 クラス - Request OFF-LINE 176
17. 1. 1 コンストラクタ 176
17. 1. 2 プロパティ 176
17. 1. 3 メソッド 176
17. 1. 3. 1 SendS1F15() 177
17. 1. 3. 2 SendS1F15_wait() 178
17. 2 class_SendS1F17 クラス - Request ON-LINE 179
17. 2. 1 コンストラクタ 179
17. 2. 2 プロパティ 179
17. 2. 3 メソッド 179
17. 2. 3. 1 SendS1F17() 180
17. 2. 3. 2 SendS1F17_wait() 181
17. 3 class_SendS2F31 クラス - Date and Time Set Requst 182
17. 3. 1 コンストラクタ 182
17. 3. 2 プロパティ 182
17. 3. 3 メソッド 182
17. 3. 3. 1 SendS2F31() 183
17. 3. 3. 2 SendS2F31_wait() 184
18. ユーザ固有 SECS-II メッセージの送信ならびに応答メッセージの送信 185

18. 1 SendReqSxFy クラス – ユーザ仕様メッセージの送信 185
18. 1. 1 コンストラクター 185
18. 1. 2 プロパテイ W 185
18. 1. 3 メソッド 185
18. 1. 3. 1 SendReuest() 186
18. 1. 3. 2 SendReuest_wait() 189
18. 2 SendResponse() – 応答メッセージの送信 191

１．はじめに

DSHEng5 GEM 通信エンジンクラス説明書は、Vol-1 から 6 までの６つの Volume に分けられています。
本説明書の Vol 番号は**4** です。

本説明書は、SECS-II メッセージの送信クラスの機能、コンストラクタ、プロパティ、メソッドなどについて説明します。

Vol 番号	文書番号	内容
Vol-1	DSHENG5-19-30321-00	エンジン起動・停止、通信確立関連クラス （ EngAPI、GEM 通信確立、予約装置変数関連）
Vol-2	DSHENG5-19-30322-00	変数情報関連クラス （EC, SV, DVVAL, CE, Report, Alarm ）
Vol-3	DSHENG5-19-30323-00	プロセス情報関連クラス （PP, FPP, RECIPE, PRJ, CJ, CARRIER, SUBSTRATE ）
Vol-4	**DSHENG5-19-30324-00**	**SECS-II メッセージ送信クラス**
Vol-5	DSHENG5-19-30325-00	SECS-II 通信メッセージ情報保存クラス
Vol-6	DSHENG5-19-30326-00	SECS-II 通信メッセージエンコード／デコード処理クラス

DSHEng5 が管理する基本的な GEM, GEM300 関連情報に関するクラスの説明については Vol-1 の説明書を参照してください。

本クラスライブラリがサポートするメッセージの一覧表を次ページに示します。

[SECS-Ⅱメッセージ一覧表]

本ライブラリがサポートするメッセージの一覧表を示します。

この表に出ていないメッセージの送受信については、17. send_request(), send_request_wait()を参照してください。

	メッセージ	クラス名	機能概略
1	S1F3,4	class_SendS1F3	S1F3 送信　Selected Equipment Status Request
		-	S1F4 応答
2	S1F11,12	class_SendS1F11	S1F11 送信　Status Variable Namelist Request
		-	S1F12 応答
3	S1F15,16	class_SendS1F15	S1F15 送信　Request OFF-LINE
		class_SendS1F16Response	S1F16 応答
4	S1F17,18	class_SendS1F17	S1F17 送信　Request ON-LINE
		class_SendS1F18Response	S1F18 応答
5	S2F13,14	class_SendS2F13	S2F13 送信　Equipment Constant Request
		-	(S2F14 はｴﾝｼﾞﾝが自動応答)
6	S2F15,16	class_SendS2F15	S2F15 送信　New Equipment Constant Send
		-	(S2F14 はｴﾝｼﾞﾝが自動応答)
7	S2F23,24	class_SendS2F23	S2F23 送信　Trace Initialize Send
		class_SendS2F24Response	S2F24 応答
8	S2F29,30	class_SendS2F29e	S2F29 送信　Equipmeny Constant Namelist Request
		-	(S2F30 はｴﾝｼﾞﾝが自動応答)
9	S2F31,32	class_SendS2F31	S2F31 送信　Date and Time Set Requst
		-	(S2F32 はｴﾝｼﾞﾝが自動応答)
10	S2F33,34	class_SendS2F33	S2F33 送信　Define Report
		-	(S2F34 はｴﾝｼﾞﾝが自動応答)
11	S2F35,36	class_SendS2F35	S2F35 送信　Link Event Report
		-	(S2F36 はｴﾝｼﾞﾝが自動応答)
12	S2F37,38	class_SendS2F37	S2F37 送信　Enable/Disabel Event Report
		-	(S2F38 はｴﾝｼﾞﾝが自動応答)
13	S2F41,42	class_SendS2F41	S2F41 送信　Host Command Send
		class_SendS2F42Response	S2F42 応答
14	S2F43,44	class_SendS2F43	S2F43 送信　Reset Spooling Stream and Function
		class_SendS2F44Response	S2F44 応答
15	S2F45,46	class_SendS2F45	S2F45 送信　Define Variable Limit Attributes
		class_SendS2F46Response	S2F46 応答
16	S2F47,48	class_SendS2F47	S2F47 送信　Variable Limit Attributes Request
		-	(S2F48 はｴﾝｼﾞﾝが自動応答)
17	S2F49,50	class_SendS2F49	S2F49 送信　Enhanced Remote Command
		class_SendS2F50Response	S2F50 応答
18	S3F17,18	class_SendS3F17	S3F17 送信　Carrier Action Request
		class_SendS3F18Response	S3F18 応答
19	S3F23,24	class_SendS3F23	S3F23 送信　Port Group Action Request
		class_SendS3F24Response	S3F24 応答

20	S3F25, 26	class_SendS3F25	S3F25 送信　Port Action Request
		class_SendS3F24Response	S3F26 応答
21	S3F27, 28	class_SendS3F27	S3F27 送信　Change Access
		class_SendS3F28Response	S3F28 応答
22	S5F1, 2	class_SendS5F1	S5F1 送信　Alarm Report Send
		class_SendS5F2Response	S5F2 応答
23	S5F3, 4	class_SendS5F3	S5F3 送信　Enable/Disabel Alarm Send
		－	(S5F4 はｴﾝｼﾞﾝが自動応答)
24	S5F5, 6	class_SendS5F5	S5F5 送信　List Alarm Request
		－	(S5F6 はｴﾝｼﾞﾝが自動応答)
25	S6F1, S6F2	－	(S6F1 はｴﾝｼﾞﾝが自動応答) Trace Data Send
		class_SendS6F2Response	S6F2 応答
26	S6F11, S6F12	class_SendS6F11	S6F11 送信　Event Report Send
		class_SendS6F12Response	S6F12 応答
27	S6F15, S6F16	class_SendS6F15	S6F15 送信　Event Report Request
		－	(S6F16 はｴﾝｼﾞﾝが自動応答)
28	S6F19, S6F20	class_SendS6F19	S6F19 送信　Indivisual Report Data
		－	(S6F20 はｴﾝｼﾞﾝが自動応答)
29	S6F23, S6F24	class_SendS6F23	S6F23 送信　Request Spooled Data
		－	(S6F24 はｴﾝｼﾞﾝが自動応答)
30	S7F1, S7F2	class_SendS7F1	S7F1 送信　Process Program Load Inquire
		class_SendS7F2Response	S7F2 応答
31	S7F3, S7F4	class_SendS7F3	S7F3 送信　Process Program Send
		class_SendS7F4Response	S7F4 応答
32	S7F5, S7F6	class_SendS7F5	S7F5 送信　Process Program Data
		class_SendS7F6Response	S7F6 応答　(通常ｴﾝｼﾞﾝ自動応答)
33	S7F17, S7F18	class_SendS7F17	S7F17 送信　Delete Process Program Send
		－	(S7F18 はｴﾝｼﾞﾝが自動応答)
34	S7F19, S7F20	class_SendS7F19	S7F19 送信　Current EPPD Request
		－	(S7F20 はｴﾝｼﾞﾝが自動応答)
35	S7F23, S7F24	class_SendS7F23	S7F23 送信　Formatted Process Program Send
		class_SendS7F24Response	S7F24 応答
36	S7F25, S7F26	class_SendS7F25	S7F25 送信　Formatted Process Program Data
		class_SendS7F26Response	S7F26 応答　(通常ｴﾝｼﾞﾝ自動応答)
37	S10F1, S10F2	class_SendS10F1	S10F1 送信　Terminal Request
		class_SendS10F2Response	S10F2 応答
38	S10F3, S10F4	class_SendS10F3	S10F3 送信　Treminal Display, Single
		class_SendS10F4Response	S10F4 応答
39	S10F5, S10F6	class_SendS10F5	S10F5 送信　Terminal Display, Multi-Block
		class_SendS10F6Response	S10F6 応答
40	S14F9, S14F10	class_SendS14F9	S14F9 送信　Create Object Request
		class_SendS14F10Response	S14F10 応答
41	S14F11, S14F12	class_SendS14F11	S14F11 送信　Delete Object Request
		class_SendS14F12Response	S14F12 応答
42	S15F3, S15F4	class_SendS15F3	S15F3 送信　Recipe Name Space Action Request
		class_SendS15F4Response	S15F4 応答

43	S15F5, S15F6	class_SendS15F5	S15F5 送信　Recipe Name space Rename Request
		class_SendS15F6Response	S15F6 応答
44	S15F7, S15F8	class_SendS15F7	S15F7 送信　Recipe Space Request
		－	(S15F8 はｴﾝｼﾞﾝが自動応答)
45	S15F9, S15F10	class_SendS15F9	S15F9 送信　Recipe Status Request
		－	(S15F10 はｴﾝｼﾞﾝが自動応答)
46	S15F13, S15F14	class_SendS15F13	S15F13 送信　Recipe Create Request
		class_SendS15F14Response	S15F14 応答
47	S15F17, S15F18	class_SendS15F17	S15F17 送信　Recipe Retrieve Request
		class_SendS15F18Response	S15F18 応答
48	S16F5, S16F6	class_SendS16F5	S16F5 送信　Process Job Command Request
		class_SendS16F6Response	S16F6 応答
49	S16F11, S16F12	class_SendS16F11	S16F11 送信　PrJobCreateEnh
		class_SendS16F12Response	S16F12 応答
50	S16F15, S16F16	class_SendS16F15	S16F15 送信　PrJobMultiCreate
		class_SendS16F16Response	S16F16 応答
51	S16F17, S16F18	class_SendS16F17	S16F17 送信　PrJobDeque
		class_SendS16F18Response	S16F18 応答
52	S16F19, S16F20	class_SendS16F19	S16F19 送信　PrGetAllJobs
		class_SendS16F20Response	S16F20 応答
53	S16F21, S16F22	class_SendS16F21	S16F21 送信　PrGetSpace
		class_SendS16F22Response	S16F22 応答
54	S16F27, S16F28	class_SendS16F27	S16F27 送信　Control Job Command Request
		class_SendS16F28Response	S16F28 応答

２．SECS-II メッセージ送受信の仕方

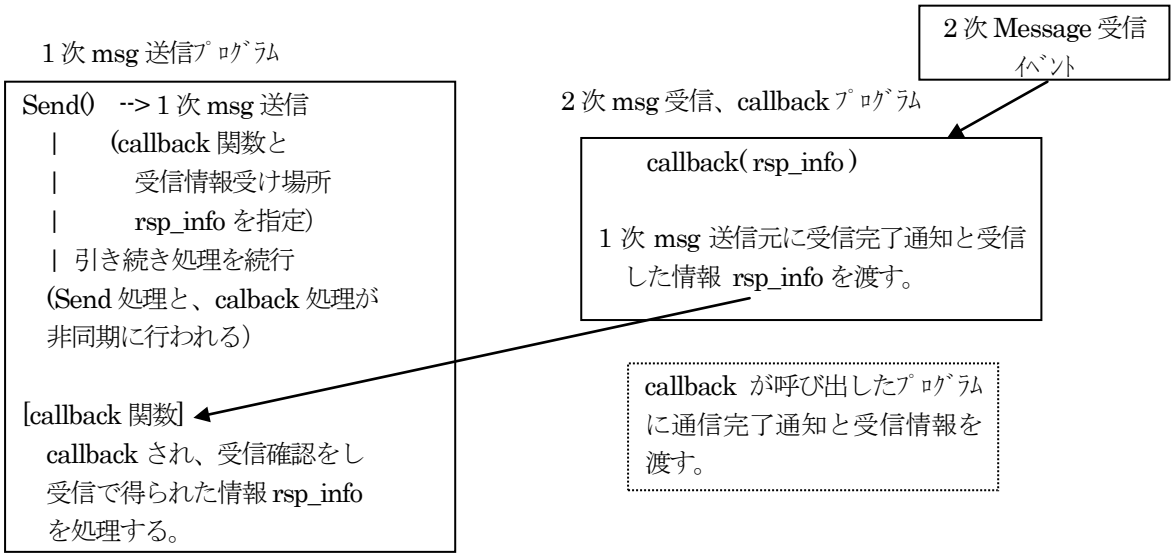
２．１　１次メッセージの送信と２次メッセージ受信

DSHEng5 通信エンジンは、SECS-II メッセージの送信方法として２つのモードを提供します。
非ブロックモード(non-block)と、ブロックモードです。

（1）非ブロックモード

APP が 1 次メッセージの送信を DSHEng5 に要求します。その際、DSHEng5 が応答メッセージの受信を報せてもらうためのコールバック(Callback)関数を引数の中に渡しておきます。そして、コントロールは応答メッセージを受信する前に APP に戻ります。

その後、DSHEng5 は応答メッセージを受信したときに、先に渡されたコールバック関数に応答メッセージ情報を付けて呼び出します。
APP は呼び出されたこのコールバック関数の中で応答メッセージ情報の処理を行います。



（2）ブロックモード

APP が 1 次メッセージの送信を DSHEng5 に要求します。その際に、応答メッセージ情報を保存してもらうための領域を指定しておきます。その後、APP は応答メッセージを受信するまで待機します。応答メッセージを受信するとコントロールが戻り、保存領域に応答情報が渡されます。

```
Send_wait()　--> 1 次 msg 送信
|
|　受信情報受け場所
|　rsp_info を指定
(受信完了まで待機-block)
<----- 受信完了
rsp_info 情報を処理して
先に進む。
```

注意
Windows Form 例えば、Button などのｺﾝﾎﾟｰﾈﾝﾄからｲﾍﾞﾝﾄﾊﾝﾄﾞﾗｰﾌの中でﾌﾞﾛｯｸﾓｰﾄﾞで送信すると、相手が応答を返すまでﾌﾟﾛｸﾞﾗﾑがﾌﾞﾛｯｸされます。もし、相手が応答できなかった場合は、T3 ﾌﾟﾛﾄｺﾙ時間を経過するまで、Windows の操作ができなくなります。
ﾌﾞﾛｯｸﾓｰﾄﾞの送信は、ｽﾚｯﾄﾞを生成し、その中で応答を受信し、処理するようにすることを推奨します。

２．１．１　非ブロックモードのプログラミング例

S1F13 については、送信のためのクラスは、class_SendS2F13 で、そのインスタンス名を Send_class とします。
変数 ID は uint 変数に設定されているものとします。

（1）送信メッセージのために用意されているクラスのインスタンスを生成します。

```
class_SendS2F13 Send_class = new class_SendS2F13();
```

（2）S2F13 に情報を要求したい ECID を TVID_LIST クラスのリストに設定します。
eng_id クラスに EC_Mdln, EC_SoftRev が定義されていたとして、この２つの ECID の情報を得たい場合は、以下のようにします。

```
TVID_LIST vid_list = new TVID_LIST();
vid_list.add( eng_id.EC_Mdln);
vid_list.add(eng_api.EC_SoftRev);
```

（3）メッセージを送信するためのメソッドを実行します。

```
int ei = class_SendS2F13.SendS2F13(vid_list, cback_S2F13, 213)
```

ここで、引数について説明します。

①vid_list
上の（2）で準備した TVID_LIST のインスタンスです。

②cback_S2F13
エンジンから終了通知を受けるためのコールバック(イベント通知)関数を指定します。

```
private static class_CALLBACK.callback_S2F13 cback_S2F13 = new
                                        class_CALLBACK.callback_S2F13(callback_S2F13);
```

(callback_S2F13 は次ページ(4)で説明します)

コールバック関数は class_CALLBACK クラス内で定義されています。
callback_S2F13 が実際のハンドラーです。これについては （4）で説明します。

③最後の 213
uint 型の引数で、ユーザが使用できるタグです。要求したプログラムが終了通知を受けたときに返却されます。ここでは、S2F13 の 213 を指定しています。

送信要求が受け付けられたかどうは、このメソッドの返却値で判断します。
返却値=0 ならば受け付けされたことを意味し、(-1)ならば受け付けられなかったことを意味します。

正常に受け付けられたら、終了イベント通知を待ちます。終了イベントは、メッセージ送信メソッドの引数 callback 関数として与えられたコールバック関数の呼び出しで通知されます。

（4）送信メソッドの引数に与える callback 関数について説明します。
（3）－③で出てきました callback_S2F13 の例は次のようになります。

```
private static int callback_S2F13( int end_status, TV_VALUE_LIST rsp_info, uint upara)
{
      EngAPI.OutLog(" ! Send_info SendS2F13 Callback() end_status = " +
                                                    end_status.ToString() + "¥n");
      EngAPI.OutLog("    upara = " + upara.ToString() + "¥n");
      if (end_status == 0)
      {
          disp_v_info.disp_TV_VALUE_LIST("----- EC value list -----", ref rsp_info);
      }
      return 0;
}
```

callback_S2F13 関数では３つの引数が付いてきます。それぞれは、次の意味になります。

①end_status　：終了状態ｺｰﾄﾞです。　=0 で正常終了、 =(-1)で異常終了を示します。
②rsp_info　　：S2F14 の情報が保存されているインスタンスで、(3)の rsp_info213 です。
③upara　　：ユーザパラメータ（タグ）です。(3)で与えた 213 の値が戻されます。

この **callback 関数は、static** にしてください。そして、0 を返却してください。

以上、S2F13 を例に説明しましたが、送信要求に使用するクラス、送信関数の引数、コールバック関数の引数は、各メッセージごとに違います。詳しい内容については、３.以降に説明します。

２．１．２　ブロックモードのプログラミング

送信のためのクラスは、class_SendS2F13 で、そのインスタンス名を Send_class とします。
装置 ID は int 変数に設定されているものとします。

（1）送信メッセージのために用意されているクラスのインスタンスを生成します。

```
class_SendS2F13 Send_class = new class_SendS2F13();
```

（2）S2F13 に情報を要求したい ECID を TVID_LIST クラスのリストに設定します。
eng_id クラスに EC_Mdln, EC_SoftRev が定義されていたとして、この２つの ECID の情報を得たい場合は、以下のようにします。

```
TVID_LIST vid_list = new TVID_LIST();
vid_list.add(eng_id.EC_Mdln);
vid_list.add(eng_api.EC_SoftRev);
```

（3）メッセージを送信するためのメソッドを実行します。

```
int ei = Send_class.SendS2F13_wait(vid_list, ref rsp_info213)
```

ここで、引数は次のとおりです。
①vid_list
　上の（2）で準備した TVID_LIST のインスタンスです。

②rsp_info213
　は受信する応答メッセージ S2F14 の内容を保存するためのクラス TV_VALUE_LIST のインスタンスです。次のように static 指定で静的変数として準備しておきます。

```
private static TV_VALUE_LIST rsp_info213 = new TV_VALUE_LIST();
```

SendS2F13_wait()によって、２次メッセージの受信、または通信エラーが発生し、終了するまでプログラムはブロックされます。（先に進みません。）

通信が正常に終了すると ei= 0 が返却されます。そして、rsp_info213 に受信した S2F14 メッセージ内に含む情報が TV_VALUE_LIST クラスのインスタンス rsp_info213 に渡されます。

この後、プログラムが rsp_info213 の処理を行い、先に進みます。

２．２　受信１次メッセージに対する２次メッセージの応答の仕方

APP プログラムでの SECS-Ⅱ1 次メッセージの受信と処理の手順は以下のようになります。

（1）EngAPI クラスを使って DHSEng5 エンジンを起動し、相手装置との通信確立を行い、そして、相手装置からの SECS-II メッセージを受信を待機します。

（2）APP は、EngAPI クラスに対して、受信した１次メッセージの中で、自分で処理したい１次メッセージを登録しておきます。
また、APP は EngAPI クラスに対して、APP が処理するメッセージを受信した際に呼び出してもらうイベントハンドラーの登録も行います。
（EngAPI.set_ev_handler()メソッドを使用します。 Vol-1　4.1.3.20　参照）

（3）DSHEng5 は、ポーリングの開始後、相手装置からメッセージを受信したとき、（2）で指定されたイベントハンドラーに、受信した SECS-II メッセージ情報の引数を付けて呼び出します。

（4）イベントハンドラーには、２つの引数が与えられます。
①DSHMSG mmsg　：メッセージ情報です。stream, function, message text とその長さです。
②trid　：HSMS 通信ドライバーが発行したトランザクション ID です。応答メッセージを返す際に、この trid を付けて通信ドライバーに渡します。

（5）APP は受けた受信メッセージの処理を行います。
処理は、メッセージ ID によって、分けて処理されることになります。
（処理方法は、デモプログラムを参考にしてください。）

（6）受信メッセージ処理を終えた後、2 次メッセージを応答するための処理になります。
DHSEng5 エンジンの下では、以下のように応答メッセージを作成し、送信します。
応答メッセージの作成し送信することになります。

（7）DSGMSG 構造体の中に応答メッセージを作成します。
メッセージ ID 別にメッセージを Encode するためのクラスが用意されており、それを使用して応答メッセージを作成します。例えば、S2F42 メッセージを作成するクラスとメソッドは以下の通りです。

```
class_S2F41.EncodeS2F42( ref DSHMSG smsg, TRCMD_ERR_INFO err_info);
err_info.Dispose();                                    // err_info の Dispose を行う。
```

（8）応答メッセージの作成ができた後、EngAPI クラスの SendResponse()メソッドでメッセージを送信します。
（7）で生成した DSHMSG　smsg の送信は次のようになります。

```
int result = EngAPI.SendResponse( ref smsg, (uint) trid)
```

（参照）18. 2　SendResponse() - 応答メッセージの送信

次ページに DSHEng5 が自動応答する場合の S2F41 の受信処理サンプルを示します。

この例では、現在実行できないとする ack = 2 にしています。

プログラムサンプル　S2F41
1受信処理

```
public static int S2F41_r_proc(ref DSHMSG smsg, int trid)
{
    int ack = 0;
    DSHMSG rmsg = new DSHMSG();
    int ei = 0;
    TRCMD_INFO info = new TRCMD_INFO();
    TRCMD_ERR_INFO erinfo = new TRCMD_ERR_INFO();

    ei = class_S2F41.DecodeS2F41(ref smsg, ref info);
    if (ei < 0) {
        ack = 1;
    }
    else
    {
        ack =2;                                          // hcack=2 :現在実行できない
    }
    erinfo.hc_ack = ack;
    class_S2F41.EncodeS2F42(ref rmsg, erinfo);

    ei = EngAPI.SendResponse(ref rmsg, (uint)trid);     // 応答ﾒｯｾｰｼﾞ送信
    info.Dispose();
    erinfo.Dispose();
    return ei;
}
```

３．装置定数(EC)関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S2F13,14	class_SendS2F13	S2F13 送信　Equipment Constant Request
		－	(S2F14 はｴﾝｼﾞﾝが自動応答)
2	S2F15,16	class_SendS2F15	S2F15 送信　New Equipment Constant Send
		－	(S2F14 はｴﾝｼﾞﾝが自動応答)
3	S2F29,30	class_SendS2F29	S2F29 送信　Equipmeny Constant Namelist Request
		－	(S2F30 はｴﾝｼﾞﾝが自動応答)

３．１　class_SendS2F13 クラス - Equipment Constant Request

S2F13 メッセージを送信し、S2F14 応答メッセージを受信するためのクラスです。
応答メッセージ S2F14 の情報は callback 関数の引数に渡されます。

３．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F13()	ｲﾝｽﾀﾝｽを生成します。

３．１．２　プロパティ

なし。

３．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F13()	非ブロックモードで送信します。
2	public int SendS2F13_wait()	ブロックモードで送信します。

[参照クラス]

	クラス名	参照場所
1	TVID_LIST	エンジンクラス説明書-Vol-2 3.2　TVID_LIST - 変数 ID 保存配列リストクラス
2	TV_VALUE_LIST	エンジンクラス説明書-Vol-2 3.4　TV_VALUE_LIST - 変数値保存配列リストクラス

３．１．３．１　SendS2F13() - 非ブロックモードでの送信

S2F13 メッセージの送信要求をします。
応答メッセージ S2F14 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F13(TVID_LIST list, class_CALLBACK.callback_S2F14 callback, UInt32 upara)

【引数】

list

ECID が保存されている配列リスト

callback

S2F13 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数 list に保存されている count 分の ECID から S2F13 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F14(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TV_VALUE_LIST list,             // S2F14 に含まれる装置定数情報
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

３．１．３．２　SendS2F13_wait() - ブロックモードでの送信

S2F13 メッセージを送信し、引き続き応答メッセージの受信も行います。

【構文】

public int SendS2F13_wait(VID_LIST list, ref TV_VALUE_LIST rsp_info)

【引数】

list

ECID が保存されている配列リスト

rsp_info

S2F14 応答メッセージに含まれる装置定数情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 list に保存されている count 分の ECID から S2F13 メッセージを生成し、それを相手装置へ送信するように要求します。
送信要求の後、応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、rsp_list に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

３．２　class_SendS2F15 クラス - New Equipment Constant Send

S2F15 メッセージを送信し、S2F16 応答メッセージを受信するためのクラスです。

３．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F15()	空のｲﾝｽﾀﾝｽを生成します。

３．２．２　プロパティ

なし。

３．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F15()	S2F15 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F15_wait()	S2F15 ﾒｯｾｰｼﾞを送信し、S2F16 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TV_VALUE_LIST	エンジンクラス説明書-Vol-2 3.5　TV_VALUE_LIST - 変数値保存配列リストクラス

３．２．３．１　SendS2F15()

S2F15 メッセージの送信要求をします。
応答メッセージ S2F16 の情報は callback 関数の引数に渡されます。

【構文】

public int Send2F15(TVID_VAL_LIST list, class_CALLBACK.callback_S2F16 callback, UInt32 upara)

【引数】

list

EC 情報が保存されている配列リスト
EC 情報には、ID, 値の format, 長さ、値が保存されているメモリが含まれます。

callback

S2F15 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数 list に保存されている count 分の ECID 情報を含む S2F15 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は、送信後、受信した応答メッセージを rsp_info のインスタンスに設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F16(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int eac,                        // S2F16 に含まれる eac Ack です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

３．３．４．２　SendS2F15_wait()

S2F15 メッセージを送信し、引き続き応答メッセージの受信も行います。

【構文】

public int SendS2F15_wait(TVID_VAL_LIST list, ref int eac)

【引数】

list

EC 情報が保存されている配列リスト
EC 情報には、ID, 値の format, 長さ、値が保存されているメモリが含まれます。

eac

S2F16 応答メッセージの eac (ack)保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 vid_list に保存されている count 分の EC 情報の内容から S2F15 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F16 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、eac に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

３．３　class_SendS2F29 クラス - Equipmeny Constant Namelist Request

S2F29 メッセージを送信し、S2F30 応答メッセージを受信するためのクラスです。

３．３．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F29()	ｲﾝｽﾀﾝｽを生成します。

３．３．２　プロパティ

なし。

３．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F29()	S2F29 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F29_wait()	S2F29 ﾒｯｾｰｼﾞを送信し、S2F30 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TVID_LIST	エンジンクラス説明書-Vol-2 3.2　TVID_LIST - 変数 ID 保存配列リストクラス
2	TEC_NAME_LIST	エンジンクラス説明書-Vol-2 3.8　TEC_NAME_LIST - EC 変数情報保存配列リストクラス

３．３．３．１　SendS2F29()

S2F29 メッセージの送信要求をします。
応答メッセージ S2F30 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F29(TVID_LIST list, class_CALLBACK.callback_S2F30 callback, UInt32 upara)

【引数】

list

ECID が保存されている配列リスト

callback

S2F29 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数 list に保存されている count 分の ECID 配列リストから S2F29 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F30(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TEC_NAME_LIST name_list,        // S2F30 に含まれる装置定数名情報　Vol-4.8 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

３．３．３．２　SendS2F29_wait()

S2F29 メッセージを送信し、引き続き応答メッセージの受信も行います。

【構文】

public int SendS2F29_wait(TVID_LIST list, ref TEC_NAME_LIST vlist)

【引数】

list

ECID が保存されている配列リスト

vlist

S2F30 応答メッセージに含まれる EC 情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 vid_list に保存されている count 分の ECID 配列リストから S2F29 メッセージを生成し、それを相手装置へ送信するように要求します。
送信要求の後、S2F30 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、rsp_list に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

４．装置状態変数(SV)関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S1F3,4	class_SendS1F3	S1F3 送信　Selected Equipment Status Request
		－	S1F4 応答
2	S1F11,12	class_SendS1F11	S1F11 送信　Status Variable Namelist Request
		－	S1F12 応答

４．１　class_SendS1F3 クラス - Selected Equipment Status Request

S1F3 メッセージを送信し、S1F4 応答メッセージを受信するためのクラスです。

４．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS1F3()	ｲﾝｽﾀﾝｽを生成します。

４．１．２　プロパティ

なし。

４．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS1F3()	S1F3 ﾒｯｾｰｼﾞを送信します。
2	public int Send_wait()	S1F3 ﾒｯｾｰｼﾞを送信し、S1F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TVID_LIST	エンジンクラス説明書-Vol-2 3.2 TVID_LIST　- 変数 ID 保存配列リストクラス
2	TV_VALUE_LIST	エンジンクラス説明書-Vol-2 3.5 TV_VALUE_LIST - 変数値保存配列リストクラス

４．１．３．１　SendS1F3()

S1F3 メッセージの送信要求をします。
応答メッセージ S1F4 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS1F3(TVID_LIST list, class_CALLBACK.callback_S1F4 callback, UInt32 upara)

【引数】

list

SVID が保存されている配列リスト

callback

S1F3 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list に保存されている count 分の SVID リストから S1F3 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージを callback 関数の引数として呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S1F4(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TV_VALUE_LIST vlist,            // S1F4 に含まれる装置状態変数情報　Vol-1 4.6 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

４．１．３．２　SendS1F3_wait()

S1F3 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS1F3_wait(TVID_LIST list, ref TV_VALUE_LIST rsp_info)

【引数】

rsp_info

S1F4 応答メッセージに含まれる装置状態変数情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 vid_list に保存されている count 分の SVID を含む S1F3 メッセージを生成し、それを相手装置へ送信するように要求します。
送信要求の後、S1F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、TV_VALUE_LIST クラスのインスタンス rsp_info に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

４．２　class_SendS1F11 クラス - Status Variable Namelist Request

S1F11 メッセージを送信し、S1F12 応答メッセージを受信するためのクラスです。

４．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS1F11()	ｲﾝｽﾀﾝｽを生成します。

４．２．２　プロパティ

なし。

４．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS1F11()	S1F11 ﾒｯｾｰｼﾞを送信します。
2	public int SendS1F11_wait()	S1F11 ﾒｯｾｰｼﾞを送信し、S1F12 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TVID_LIST	エンジンクラス説明書-Vol-2 3.2　TVID_LIST - 変数 ID 保存配列リストクラス

４．２．３．１　SendS1F11()

S1F11 メッセージの送信要求をします。
応答メッセージ S1F12 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS1F11(TVID_LIST list, class_CALLBACK.callback_S1F11 callback, uint upara)

【引数】

list

SVID が保存されている配列リスト。

callback

S1F11 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list に保存されている count 分の SVID を含む S1F11 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S1F12(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TSV_NAME_LIST rsp_list,         // S1F12 に含まれる装置状態変数名情報　Vol-4.9 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

４．２．３．２　SendS1F11_wait()

S1F11 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS1F11_wait(TVID_LIST list, ref TSV_NAME_LIST rsp_list)

【引数】

TVID_LIST list,

rsp_list

S1F12 応答メッセージに含まれる名前情報を保存するためのクラスのインスタンスです

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど）

【説明】

引数 vid_list に保存されている count 分の SVID を含む S1F11 メッセージを生成し、それを相手装置へ送信するように要求します

送信要求の後、S1F12 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。

応答情報は、TSV_NameList クラスのインスタンス rsp_list に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

５．変数リミット(LIMIT)関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S2F45,46	class_SendS2F45	S2F45 送信　Define Variable Limit Attributes
		class_SendS2F46Response	S2F46 応答
2	S2F47,48	class_SendS2F47	S2F47 送信　Variable Limit Attributes Request
		－	(S2F48 はｴﾝｼﾞﾝが自動応答)

５．１　class_SendS2F45 クラス - Define Variable Limit Attributes

S2F45 メッセージを送信し、S2F46 応答メッセージを受信するためのクラスです。

５．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F45()	ｲﾝｽﾀﾝｽを生成します。

５．１．２　プロパティ

なし。

５．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F45()	S2F45 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F45_wait()	S2F45 ﾒｯｾｰｼﾞを送信し、S2F46 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TLIMIT_LIST	エンジンクラス説明書-Vol-2 3.12　TLIMIT_LIST - 装置変数リミット情報リストクラス
2	TLIMIT_ERR_LIST	

５．１．３．１　SendS2F45()

S2F45 メッセージの送信要求をします。
応答メッセージ S2F46 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F45(TLIMIT_LIST info, class_CALLBACK.callback_S2F46 callback, UInt32 upara)

【引数】

info

TLIMIT_LIST クラスのインスタンスで、送信したいリミット情報です。

callback

S2F45 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 limit_list に保存されている count 分の vid の変数リミットから S2F45 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F46(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TLIMIT_ERR_LIST rsp_info,       // S2F46 に含まれる変数ﾘﾐｯﾄ設定応答情報 Vol-1 5.4 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

５．１．３．２　SendS2F45_wait()

S2F45 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F45_wait(TLIMIT_LIST info, ref TLIMIT_ERR_LIST erinfo)

【引数】

info

TLIMIT_LIST クラスのインスタンスで、送信したいリミット情報です。

err_info

S2F46 応答メッセージに含まれる変数リミット設定応答情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 info に保存されている count 分の vid の変数リミットから S2F45 メッセージを生成し、それを相手装置へ送信するように要求します。
送信要求の後、S2F46 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、TLIMIT_ERR_LIST クラスのインスタンス erinfo に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

５．２　class_SendS2F47 クラス - Variable Limit Attributes Request

S2F47 メッセージを送信し、S2F48 応答メッセージを受信するためのクラスです。

５．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F47()	ｲﾝｽﾀﾝｽを生成します。

５．２．２　プロパティ

なし。

５．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F47()	S2F47 ﾒｯｾｰｼﾞを送信します。
2	public int Send2F47_wait()	S2F47 ﾒｯｾｰｼﾞを送信し、S2F48 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TVID_LIST	エンジンクラス説明書-Vol-2 3.2 TVID_LIST　- 変数 ID 保存配列リストクラス
2	TLIMIT_RSP_LIST	

５．２．３．１　SendS2F47()

S2F47 メッセージの送信要求をします。
応答メッセージ S2F47 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F47(TVID_LIST vid_list, class_CALLBACK.callback_S2F48 callback, UInt32 upara)

【引数】

vid_list

VID 配列リストです。

callback

S2F47 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 vid_list に保存されている count 分の変数 ID を含む S2F47 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージを rsp_info のインスタンスに設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F48(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TLIMIT_RSP_LIST rsp_info,       // S2F48 に含まれる変数リミット名情報　 Vol-5.5 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

５．２．３．２　SendS2F47_wait()

S2F47 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(TVID_LIST vid_list, TLIMIT_RSP_LIST rsp_info)

【引数】

vid_list

VID 配列リストです。

rsp_info

S2F48 応答メッセージに含まれる変数リミット情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 vid_list に保存されている count 分の変数 ID を含む S2F47 メッセージを生成し、それを相手装置へ送信するように要求します。
送信要求の後、S2F48 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、TLIMIT_RSP_LIST クラスのインスタンス rsp_info に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

６．トレース(TRACE)関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S2F23,24	class_SendS2F23	S2F23 送信　Trace Initialize Send
		class_SendS2F24Response	S2F24 応答
2	S6F1,S6F2	－	(S6F1 はｴﾝｼﾞﾝが自動送信) Trace Data Send
		class_SendS6F2Response	S6F2 応答

６．１　class_SendS2F23 クラス - Trace Initialize Send

S2F23 メッセージを送信し、S2F24 応答メッセージを受信するためのクラスです。

６．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F23()	ｲﾝｽﾀﾝｽを生成します。

６．１．２　プロパティ

なし。

６．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F23()	S2F23 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F23_wait()	S2F23 ﾒｯｾｰｼﾞを送信し、S2F24 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

６．１．３．１　SendS2F23()

S2F23 メッセージの送信要求をします。
応答メッセージ S2F24 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F23(TTRACE_INFO info, class_CALLBACK.callback_S2F24 callback, UInt32 upara)

【引数】

info

トレース情報 TTRACE_INFO クラスのインスタンス

callback

S2F23 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 tr_info に保存されているトレース情報から S2F23 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージの tiaack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F24(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int tiaack,                     // S2F24 に含まれる tiaack です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

６．１．３．２　SendS2F23_wait()

S2F23 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F23_wait(TTRACE_INFO info, ref int tiaack)

【引数】

info

トレース情報が保存されているインスタンスです。

tiaack

S2F24 応答メッセージに含まれる ACK 保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 tr_info に保存されているトレース情報から S2F23 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F24 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。

応答情報は、ACK を tiaack に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

7．収集イベントとレポート関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S2F35,36	class_SendS2F35	S2F35 送信　Link Event Report
		－	(S2F36 はｴﾝｼﾞﾝが自動応答)
2	S2F37,38	class_SendS2F37	S2F37 送信　Enable/Disabel Event Report
		－	(S2F38 はｴﾝｼﾞﾝが自動応答)
3	S2F33,34	class_SendS2F33	S2F33 送信　Define Report
		－	(S2F34 はｴﾝｼﾞﾝが自動応答)
4	S6F11,S6F12	class_SendS6F11	S6F11 送信　Event Report Send
		class_SendS6F12Response	S6F12 応答
5	S6F15,S6F16	class_SendS6F15	S6F15 送信　Event Report Request
		－	(S6F16 はｴﾝｼﾞﾝが自動応答)
6	S6F19,S6F20	class_SendS6F19	S6F19 送信　Indivisual Report Data
		－	(S6F20 はｴﾝｼﾞﾝが自動応答)

７．１　class_SendS2F35 クラス - Link Event Report

S2F35 メッセージを送信し、S2F36 応答メッセージを受信するためのクラスです。

７．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F35()	ｲﾝｽﾀﾝｽを生成します。

７．１．２　プロパティ

なし。

７．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F35()	S2F35 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F35_wait()	S2F35 ﾒｯｾｰｼﾞを送信し、S2F36 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

７．１．３．１　SendS2F35()

S2F35 メッセージの送信要求をします。
応答メッセージ S2F36 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F35(TCE_LIST list, class_CALLBACK.callback_ack callback, UInt32 upara)

【引数】

list

CE リンク情報のリストです。

callback

S2F35 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list 配列に保存されているイベントリンク情報から S2F35 メッセージを生成し、それを相手装置へ送信するように要求します。
引数 list のプロパティ count=0 の場合、全 CEID のレポートの解除になります。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの lrack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F36(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int lrack,                      // S2F36 に含まれる lrack です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

７．１．３．２　SendS2F35_wait()

S2F35 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F35_wait(TCE_LIST list, ref int lrack)

【引数】

list

CE リンク情報のリストです。

lrack

S2F36 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 list 配列に保存されているイベントリンク情報から S2F35 メッセージを生成し、それを相手装置へ送信するように要求します。
引数 list のプロパティ count=0 の場合、全 CEID のレポートの解除になります。

送信要求の後、S2F36 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、ACK を lrack に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

７．２　class_SendS2F37 クラス - Enable/Disabel Event Report

S2F37 メッセージを送信し、S2F38 応答メッセージを受信するためのクラスです。
CEID を指定して Enable/Disable の設定を行います。

７．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F37()	ｲﾝｽﾀﾝｽを生成します。

７．２．２　プロパティ

なし。

７．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F37()	S2F37 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F37_wait()	S2F37 ﾒｯｾｰｼﾞを送信し、S2F38 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

７．２．３．１　SendS2F37()

S2F37 メッセージの送信要求をします。

【構文】

public int SendS2F37(TEDER_INFO info, class_CALLBACK.callback_S2F38 callback, UInt32 upara)

【引数】

info

ceed(CE Enable/Disable)情報と CEID 配列リストです。

callback

S2F37 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info に含まれる CEID リストと ceed(送信 Enable/Disable)の情報から S2F37 メッセージを生成し、それを相手装置へ送信するように要求します。
引数 info のプロパティ count=0 の場合は、全 CEID が対象になります。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答メッセージの erack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F38(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int erack,                      // S2F38 に含まれる erack です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

７．２．３．２　SendS2F37_wait()

S2F37 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F37_wait(EDER_INFO info, ref int erack)

【引数】

info

ceed(CE Enable/Disable)情報と CEID 配列リストです。

erack

S2F38 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 info の ceed と id_list 配列に保存されているイベント ID 情報から S2F37 メッセージを生成し、それを相手装置へ送信するように要求します。
引数 info のプロパティ count=0 の場合は、全 CEID が対象になります。

送信要求の後、S2F38 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、ACK を erack に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

７．３　class_SendS2F33 クラス - Define Report

S2F33 メッセージを送信し、S2F34 応答メッセージを受信するためのクラスです。

７．３．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F33()	ｲﾝｽﾀﾝｽを生成します。

７．３．２　プロパティ

なし。

７．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F33()	S2F33 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F33_wait()	S2F33 ﾒｯｾｰｼﾞを送信し、S2F34 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRP_LIST	エンジンクラス説明書-Vol-2 5.3 TRP_LIST - レポートリンク情報リスト

７．３．３．１　SendS2F33()

S2F33 メッセージの送信要求をします。
応答メッセージ S2F34 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F33(TRP_LIST info, class_CALLBACK.callback_S2F34 callback, UInt32 upara)

【引数】

info

レポート情報リストです。

callback

S2F33 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info のレポートリンク情報から S2F33 メッセージを生成し、それを相手装置へ送信するように要求します。
info のプロパティ count=0 の場合は、全レポート ID の削除指定になります。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの drack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F34(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int drack,                      // S2F34 に含まれる drack です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

７．３．３．２　SendS2F33_wait()

S2F33 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send2F33_wait(TRP_LIST info, ref int drack)

【引数】

info

　　レポート情報リストです。

drack

　　S2F34 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど）

【説明】

引数 info に保存されているレポートリンク情報から S2F33 メッセージを生成し、それを相手装置へ送信するように要求します。
info のプロパティ count=0 の場合は、全レポート ID の削除指定になります。

送信要求の後、S2F34 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。
応答情報は、ACK を drack に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

７．４　class_SendS6F11 クラス - Event Report Send

S6F11 メッセージを送信し、S6F12 応答メッセージを受信するためのクラスです。
イベントレポートを送信します。

７．４．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS6F11()	ｲﾝｽﾀﾝｽを生成します。

７．４．２　プロパティ

なし。

７．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS6F11()	S6F11 ﾒｯｾｰｼﾞを送信します。
2	public int SendS6F11_wait()	S6F11 ﾒｯｾｰｼﾞを送信し、S6F12 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

７．４．３．１　SendS6F11()

S6F11 メッセージの送信要求をします。
応答メッセージ S6F12 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS6F11(uint ceid, class_CALLBACK.callback_S6F11 callback, uint upara)

【引数】

ceid

送信するイベント ID です。

callback

S6F11 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
300	ceed=0(Enable でない)のため送信できなかった。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 ceid のレポート情報をエンジンから取得し、S6F11 メッセージを生成し、それを相手装置へ送信するように要求します。
エンジンからは、ceid にリンクされたレポートにリンクされている変数 ID の現在の変数値情報を取得し、それを、S6F11 に組み込んだ上で S6F11 メッセージを送信します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後、受信した応答結果は引数 end_status に設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S6F12(
    int end_status,                    // 終了状態ｺｰﾄﾞ
    uint upara                         // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了し，S6F12 の ack=0 であった。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。(ackc6 が 0 でなかった。)

７．４．３．２　SendS6F11_wait()

S6F11 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(uint ceid)

【引数】

ceid

送信するイベント ID です。

【戻り値】

返却値	意　味
0～255	正常に送受信した。 ackc6 の値になります。
300	ceed=0(Enable でない)のため送信できなかった。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 ceid のレポート情報をエンジンから取得し、S6F11 メッセージを生成し、それを相手装置へ送信するように要求します。
エンジンからは、ceid にリンクされたレポートにリンクされている変数 ID の現在の変数値情報を取得し、それを、S6F11 に組み込んだ上で S6F11 メッセージを送信します。

送信要求の後、S6F12 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei >= 0 の場合は、ackc6 の値が渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

７．５　class_SendS6F15 クラス － Event Report Request

S6F15 メッセージを送信し、S6F16 応答メッセージを受信するためのクラスです。
相手装置にイベントレポートの応答を要求します。

７．５．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS6F15()	ｲﾝｽﾀﾝｽを生成します。

７．５．２　プロパティ

なし。

７．５．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS6F15()	S6F15 ﾒｯｾｰｼﾞを送信します。
2	public int SendS6F15_wait()	S6F15 ﾒｯｾｰｼﾞを送信し、S6F16 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

［参照クラス］

	クラス名	参照場所
1	TCE_CONTENT	エンジンクラス説明書-Vol-2 4.6 TCE_CONTENT － CE のリンク・レポートの詳細情報保存リスト

７．５．３．１　SendS6F15()

S6F15 メッセージの送信要求をします。
応答メッセージ S6F16 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS6F15(uint ceid, class_CALLBACK.callback_S6F15 callback, uint upara)

【引数】

ceid

レポート情報を求める対象のイベント ID です。

callback

S6F15 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 ceid のレポート情報の応答を S6F15 によって相手装置に要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
DSHEng5 は送信後､受信した応答メッセージのレポート情報格納用インスタンス rsp_info を引数にして callback 関数を呼び出します。callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S6F16(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TCE_CONTENT rsp_info,           // S6F16 に含まれるﾚﾎﾟｰﾄ情報です。 Vol-2 4.3 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

７．５．３．２　SendS6F15_wait()

S6F15 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(uint ceid, ref TCE_CONTENT rsp_info)

【引数】

ceid

　　レポート情報を求める対象のイベント ID です。

rsp_info

　　S6F16 応答メッセージに含まれるレポート情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 ceid のレポート情報の応答を S6F15 によって相手装置に要求します。

送信要求の後、S6F16 応答メッセージを待機します。受信が終了したら、上の【戻り値】受信した応答メッセージのレポート情報は引数の rsp_info に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

７．６　class_SendS6F19 クラス - Indivisual Report Data

S6F19 メッセージを送信し、S6F20 応答メッセージを受信するためのクラスです。
相手装置に指定レポート ID に対するレポート情報の応答を要求します。

７．６．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS6F19()	ｲﾝｽﾀﾝｽを生成します。

７．６．２　プロパティ

なし。

７．６．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS6F19()	S6F19 ﾒｯｾｰｼﾞを送信します。
2	public int SendS6F19_wait()	S6F19 ﾒｯｾｰｼﾞを送信し、S6F20 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

７．６．３．１　SendS6F19()

S6F19 メッセージの送信要求をします。

【構文】

public int SendS6F19(UInt32 rpid, class_CALLBACK.callback_S6F20 callback, UInt32 upara)

【引数】

rpid

レポート情報を要求する対象のレポート ID です。

callback

S6F19 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rpid のレポート情報の応答を S6F19 で相手装置に要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージのレポート情報格納用インスタンス rsp_info を引数にして callback 関数を呼び出します。callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S6F20 (
    int end_status,                  // 終了状態ｺｰﾄﾞ
    TRP_CONTENT rsinfo,              // S6F20 に含まれるﾚﾎﾟｰﾄ情報です。 Vol-1  8.5 参照
    uint upara                       // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

７．６．３．２　SendS6F19_wait()

S6F19 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS6F19_wait(uint rpid, ref TRP_CONTENT rsp_info)

【引数】

rpid

レポート情報を要求する対象のレポート ID です。

rsp_info

S6F20 応答メッセージに含まれるレポート情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 rpid のレポート情報の応答を S6F19 によって相手装置に要求します。

送信要求の後、S6F20 応答メッセージを待機します。受信が終了したら、上の【戻り値】受信した応答メッセージのレポート情報は引数 rsp_info に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

８．アラーム関連メッセージの送受信

以下のメッセージがあります。

	メッセージ	クラス	機能概略
1	S5F1,2	class_SendS5F1	S5F1 送信　　Alarm Report Send
		-	
2	S5F3,4	class_SendS5F3	S5F3 送信　　Enable/Disabel Alarm Send
		-	(S5F4 はｴﾝｼﾞﾝが自動応答)
3	S5F5,6	class_SendS5F5	S5F5 送信　　List Alarm Request
		-	(S5F6 はｴﾝｼﾞﾝが自動応答)

８．１　class_SendS5F1 クラス - Alarm Report Send

S5F1 メッセージを送信し、S5F2 応答メッセージを受信するためのクラスです。
アラームレポートを送信します。

８．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS5F1()	ｲﾝｽﾀﾝｽを生成します。

８．１．２　プロパティ

なし。

８．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS5F1()	S5F1 ﾒｯｾｰｼﾞを送信します。
2	public int SendS5F1_wait()	S5F1 ﾒｯｾｰｼﾞを送信し、S5F2 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

８．１．３．１　SendS5F10

S5F1 メッセージの送信要求をします。
応答メッセージ S5F2 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS5F1(uint alid, int on_off, class_CALLBACK.callback_S5F2 callback, UInt32 upara)

【引数】

alid

アラーム ID です。

int on_off

アラーム発生／復旧の指定を行います。 1=発生、0=復旧です。

callback

S5F1 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
300	aled=0(Enable でない)のため送信できなかった。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 alid のアラーム情報をエンジンから取得し、S5F1 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答結果は引数 end_status に設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S5F2(
    int end_status,                    // 終了状態ｺｰﾄﾞ
    uint upara                         // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

８．１．３．２　SendS5F1_wait()

S5F1 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS5F1_wait(uint alid, int on_off, ref int ack5)

【引数】

alid

　　アラーム ID です。

on_off

　　アラーム発生／復旧の指定を行います。 1=発生、0=復旧です。

ackc5

　　応答コードです。

【戻り値】

返却値	意　味
0～255	正常に送受信した。 ackc5 の値になります。
300	aled=0(Enable でない)のため送信できなかった。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 alid のアラーム情報をエンジンから取得し、それに alflag を設定、S5F1 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S5F2 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei >= 0 の場合は、ackc5 の値が渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

８．２　class_SendS5F3 クラス - Enable/Disabel Alarm Send

S5F3 メッセージを送信し、S5F4 応答メッセージを受信するためのクラスです。

８．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS5F3()	ｲﾝｽﾀﾝｽを生成します。

８．２．２　プロパティ

なし。

８．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS5F3()	S5F3 ﾒｯｾｰｼﾞを送信します。
2	public int SendS5F3_wait()	S5F3 ﾒｯｾｰｼﾞを送信し、S5F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TAL_S5F3_INFO	エンジンクラス説明書-Vol-2 6.3 TAL_S5F3_INFO - アラーム有効/無効設定情報保存クラス

８．２．３．１　SendS5F30

S5F3 メッセージの送信要求をします。
応答メッセージ S5F4 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS5F3(TAL_S5F3_INFO info, class_CALLBACK.callback_S5F4 callback, UInt32 upara)

【引数】

info

アラーム有効ビット(enabled)とアラーム ID リスト情報が含まれています。

callback

S5F3 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info にに従って S5F3 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ackc5 を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S5F4(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc5,                      // S5F4 に含まれる ackc5 です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

８．２．３．２　SendS5F3_wait()

S5F3 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS5F3_wait(TAL_S5F3_INFO info, ref int ack)

【引数】

info

アラーム有効ビット(enabled)とアラーム ID リスト情報が含まれています。

ackc5

S5F2 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 info に従って S5F3 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S5F4 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、ackc5 に ACK の値が渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

８．３　class_SendS5F5 クラス - List Alarm Reques

S5F5 メッセージを送信し、S5F6 応答メッセージを受信するためのクラスです。

８．３．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS5F5()	ｲﾝｽﾀﾝｽを生成します。

８．３．２　プロパティ

なし。

８．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS5F5()	S5F5 ﾒｯｾｰｼﾞを送信します。
2	public int SendS5F5_wait()	S5F5 ﾒｯｾｰｼﾞを送信し、S5F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TAL_S5F6_INFO	エンジンクラス説明書-Vol-2 6.4 TAL_S5F6_INFO - アラーム情報保存クラス

８．３．３．１　SendS5F5()

S5F5 メッセージの送信要求をします。
応答メッセージ S5F6 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS5F5(TAL_S5F5_INFO info, class_CALLBACK.callback_S5F6 callback, UInt32 upara)

【引数】

info

アラーム ID の配列リストです。

callback

S5F5 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info の中の alid_list に保存されている count 分のアラーム ID を含む S5F5 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージを rsp_info のインスタンスに設定し、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S5F6(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TAL_S5F6_INFO rsp_info,         // S5F6 に含まれるアラーム情報　 Vol-9.4 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

８．３．３．２　SendS5F5_wait()

S5F5 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS5F5_wait(TAL_S5F5_INFO info, ref TAL_S5F6_LIST list)

【引数】

info

アラーム ID の配列リストです。

rsp_info

S5F6 応答メッセージに含まれる名前情報を保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 infot から S5F5 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S5F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージを rsp_info のインスタンスに渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．プロセスプログラム関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス	機能概略
1	S7F1, S7F2	class_SendS7F1	S7F1 送信　Process Program Load Inquire
		class_SendS7F2Response	S7F2 応答
2	S7F3, S7F4	class_SendS7F3	S7F3 送信　Process Program Send
		class_SendS7F4Response	S7F4 応答
3	S7F5, S7F6	class_SendS7F5	S7F5 送信　Process Program Data
		－	(S7F6 はｴﾝｼﾞﾝが自動応答)
4	S7F17, S7F18	class_SendS7F17	S7F17 送信　Delete Process Program Send
		－	(S7F18 はｴﾝｼﾞﾝが自動応答)
5	S7F19, S7F20	class_SendS7F19	S7F19 送信　Current EPPD Request
		－	(S7F20 はｴﾝｼﾞﾝが自動応答)

９．１　class_SendS7F1 クラス - Process Program Load Inquire

S7F1 メッセージを送信し、S7F2 応答メッセージを受信するためのクラスです。
プロセスプログラム（PP）のロード問合せ情報を送信します。

９．１．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F1()	空のｲﾝｽﾀﾝｽを生成します。

９．１．２　プロパティ

なし。

９．１．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F1()	S7F1 ﾒｯｾｰｼﾞを送信します。
2	public int SendS7F1_wait()	S7F1 ﾒｯｾｰｼﾞを送信し、S7F2 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

９．１．３．１　SendS7F1()

S7F1 メッセージの送信要求をします。
応答メッセージ S7F2 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F1(string ppid, uint length, class_CALLBACK.callback_S7F2 callback, UInt32 upara)

【引数】

ppid
　　プロセスプログラム ID です。
length
　　PP 情報ロードに必要なメモリです。
callback
　　S7F1 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数に指定された ppid, length 情報から S7F1 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ppgnt を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F2(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ppgnt,                      // S7F2 に含まれる ACK です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．１．３．２　SendS7F1_wait()

S7F1 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS7F1_wait(string ppid, uint length, ref int ack7)

【引数】

ppid

　　プロセスプログラム ID です。

length

　　情報ロードに必要なメモリです。

ppgnt

　　S7F2 応答メッセージに含まれる ACK を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数に指定された ppid, length から S7F1 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S7F2 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、ACK を ppgnt に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．２　class_SendS7F3 クラス － Process Program Send

S7F3 メッセージを送信し、S7F4 応答メッセージを受信するためのクラスです。
プロセスプログラム（PP）情報を送信します。

９．２．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F3()	空のｲﾝｽﾀﾝｽを生成します。

９．２．２　プロパティ

なし。

９．２．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F3()	S7F3 ﾒｯｾｰｼﾞを送信します。
2	public int SendS7F3_wait()	S7F3 ﾒｯｾｰｼﾞを送信し、S7F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

９．２．３．１　SendS7F30

S7F3 メッセージの送信要求をします。

【構文】

public int SendS7F3(string ppid, class_CALLBACK.callback_S7F4 callback, UInt32 upara)

【引数】

ppid

PPID　プロセスプログラム ID です。

callback

S7F3 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	ppid が登録されていなかった。 エンコードに失敗した。または、送信要求が受け入れられなかった。

【説明】

引数の ppid から S7F3 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ackc7 を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F4(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc7,                      // S7F4 に含まれる ACK です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．２．３．２　SendS7F3_wait()

S7F3 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(string ppid, ref int ackc7)

【引数】

ppid

PPID　プロセスプログラム ID です。

ackc7

S7F4 応答メッセージに含まれる ACK を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 ppid から S7F3 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S7F4 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、ACK を ackc7 に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．３　class_SendS7F5 クラス - Process Program Data

S7F5 メッセージを送信し、S7F6 応答メッセージを受信するためのクラスです。
相手装置にプロセスプログラム（PP）情報の応答を要求します。

９．３．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F5()	ｲﾝｽﾀﾝｽを生成します。

９．３．２　プロパティ

なし。

９．３．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F5()	S7F5 ﾒｯｾｰｼﾞを送信します。
2	public int Send_wait()	S7F5 ﾒｯｾｰｼﾞを送信し、S7F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPP_INFO	エンジンクラス説明書-Vol-3 2.2 TPP_INFO - プロセス・プログラム情報保存クラス

９．３．３．１　SendS7F5()

S7F5 メッセージの送信要求をします。
応答メッセージ S7F6 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F5(string ppid, class_CALLBACK.callback_S7F6 callback, uint upara)

【引数】

ppid
　　プロセスプログラム ID です。
callback
　　S7F5 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 ppid から S7F5 を送信し、プロセスプログラム情報の応答を要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後､受信した応答メッセージに含まれるプロセスプログラム情報を TPP_INFO クラスのインスタンスにデコードし、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F6(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TPP_INFO rsp_info,              // S7F6 に含まれるﾚﾎﾟｰﾄ情報です。 Vol-1  10.2 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．３．３．２　SendS7F5_wait()

S7F5 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(string ppid, ref TPP_INFO rsp_info)

【引数】

ppid

　　プロセスプログラム ID です。

rsp_info

　　S7F6 応答メッセージに含まれる PP 情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど）

【説明】

指定された ppid のプロセスプログラム情報の応答を S7F5 の送信によって相手装置に要求します。

送信要求の後、S7F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、S7F6 に含まれるプロセスプログラム情報を rsp_info に設定し渡します。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．４　class_SendS7F17 クラス - Delete Process Program Send

S7F17 メッセージを送信し、S7F18 応答メッセージを受信するためのクラスです。
S7F17 は相手装置にプロセスプログラム ID(PPID)を指定してそれらの削除を要求します。
そして応答メッセージを受信します。

９．４．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F17()	ｲﾝｽﾀﾝｽを生成します。

９．４．２　プロパティ

なし。

９．４．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F17()	S7F17 ﾒｯｾｰｼﾞを送信します。
2	public int SendS7F17_wait()	S7F17 ﾒｯｾｰｼﾞを送信し、S7F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

９．４．３．１　SendS7F17()

S7F17 メッセージの送信要求をします。
応答メッセージ S7F18 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F17(TPPID_LIST list, class_CALLBACK.callback_S7F18 callback, uint upara)

【引数】

list

送信する PPID が保存されている配列リストです。

callback

S7F17 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list 配列に含まれる PPID から S7F17 を生成し相手装置に送信要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージに含まれる ACKC7 を ackc7 に設定し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F18(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc7,                      // S7F18 に含まれる ACKC7 情報です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．４．３．２　SendS1F17_wait()

S7F17 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS7F17_wait(TPPID_LIST list, ref int ack7)

【引数】

list

送信する PPID が保存されている配列リストです。

ackc7

S7F18 応答メッセージに含まれる ACK を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 list 配列に含まれる PPID から S7F17 を生成し相手装置に送信要求します。

送信要求の後、S7F18 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、ACK を ackc7 に渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

９．５　class_SendS7F19 クラス - Current EPPD Request

S7F19 メッセージを送信し、S7F20 応答メッセージを受信するためのクラスです。
S7F19 は相手装置から全プロセスプログラム ID(PPID)の送信を要求します。

９．５．１　コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F19()	ｲﾝｽﾀﾝｽを生成します。

９．５．２　プロパティ

なし。

９．５．３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F19()	S7F19 ﾒｯｾｰｼﾞを送信します。
2	public int SendS7F19_wait()	S7F19 ﾒｯｾｰｼﾞを送信し、S7F20 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

９．５．３．１　SendS7F19()

S7F19 メッセージの送信要求をします。
応答メッセージ S7F20 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F19(class_CALLBACK.callback_S7F20 callback, uint upara)

【引数】

callback

S7F19 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

S7F19 送信を要求します。(S7F19 メッセージは header のみ)
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージに含まれる全 PPID を id_list に設定し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F20(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TPPID_LIST id_list,             // S7F20 に含まれる PPID 情報です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

９．５．３．２　SendS7F19_wait()

S7F19 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS7F19_wait(ref TPPID_LIST id_list)

【引数】

id_list

S7F20 応答メッセージで与えられる全 PPID を保存するための TPPID_LIST クラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

S7F19 送信を要求します。

送信要求の後、S7F20 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる全 PPID を id_list に設定し渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. フォーマット付きプロセスプログラム関連メッセージの送受信

以下のメッセージとクラスがあります。

1	S7F23, S7F24	class_SendS7F23	S7F23 送信　　Formatted Process Program Send
		class_SendS7F24Response	S7F24 応答
2	S7F25, S7F26	class_SendS7F25	S7F25 送信　　Formatted Process Program Data
		－	(S7F26 はｴﾝｼﾞﾝが自動応答するので準備していません。)

10. 1 class_SendS7F23 クラス- Formatted Process Program Send

S7F23 メッセージを送信し、S7F24 応答メッセージを受信するためのクラスです。
S7F23 はフォーマット付きプロセスプログラム（FPP）情報を送信するメッセージです。

10. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F23()	空のｲﾝｽﾀﾝｽを生成します。

10. 1. 2 プロパティ

なし。

10. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F23()	S7F23 ﾒｯｾｰｼﾞを送信し、S7F24 応答ﾒｯｾｰｼﾞを受信します。応答はｺｰﾙﾊﾞｯｸで受け取ります。
2	public int SendS7F23_wait()	S7F23 ﾒｯｾｰｼﾞを送信し、S7F24 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TFPP_INFO	エンジンクラス説明書-Vol-3 3.2 TFPP_INFO - 書式付きプロセス・プログラム情報保存クラス

10．1．3．1　SendS7F23()

S7F23 メッセージの送信要求をします。
応答メッセージ S7F24 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F23(string fppid, class_CALLBACK.callback_S7F24 callback, UInt32 upara)
public int SendS7F23(TFPP_INFO info, class_CALLBACK.callback_S7F24 callback, UInt32 upara)

【引数】

fppid
　　FPP ID です。
info
　　FPP 情報が保存されている TFPP_INFO クラスのインスタンスです。
callback
　　S7F23 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	fppid が登録されていなかった。 エンコードに失敗した。または要求が受け入れられなかった。

【説明】

1 番目の引数が fppid の場合は、DSHEng5 に登録されている FPP 情報を取得して処理を進めます。
引数 info に保存されている FPP 情報から S7F23 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
fppid が指定された場合は、エンジンから FPP 情報を fpp_class プロパティに取得した上で送信します。
送信後、受信した応答メッセージの ackc7 を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F24(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc7,                      // S7F24 に含まれる ACK です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．1．3．2　SendS7F23_wait()

S7F23 メッセージを送信し、引き続き応答メッセージの受信も行います。

【構文】

public int SendS7F23_wait(string fppid, ref int ack7)
public int SendS7F23_wait(TFPP_INFO info, ref int ack7)

【引数】

fppid

FPPID です。

info

FPP 情報が保存されている TFPP_INFO クラスのインスタンスです。

ackc7

S7F24 応答メッセージに含まれる ACK を保存します。

【戻り値】

返却値	意　味
0	正常に送信し、応答 ackc7=0 であった。
(-1)	fppid が登録されていなかった。 エンコードに失敗した。または要求が受け入れられなかった。

【説明】

引数に fppid が与えられた場合には、エンジンから FPP 情報を fpp_class 内に取得します。
引数に与えられない場合は、予め、fpp_class の fppid, mdln, softrev, プロセスコマンド情報が引数 info に設定されていなければなりません。
送信要求の後、S7F24 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

10. ２ class_SendS7F25 クラス - Formatted Process Program Data

S7F25 メッセージを送信し、S7F26 応答メッセージを受信するためのクラスです。
相手装置にフォーマット付きプロセスプログラム（FPP）情報の応答を要求します。

10. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS7F25()	ｲﾝｽﾀﾝｽを生成します。

10. ２. ２ プロパティ

なし。

10. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS7F25()	S7F25 ﾒｯｾｰｼﾞを送信します。 応答はｺｰﾙﾊﾞｯｸ関数で渡されます。
2	public int SendS7F25_wait()	S7F25 ﾒｯｾｰｼﾞを送信し、S7F26 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

10. 2. 3. 1 SendS7F25()

S7F25 メッセージの送信要求をします。
応答メッセージ S7F26 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS7F25(string fppid, class_CALLBACK.callback_S7F26 callback, uint upara)

【引数】

fppid

FPP ID です。

callback

S7F25 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意 味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 fppid から S7F25 を送信し、FPP 情報を要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージに含まれるプロセスプログラム情報格納用インスタンス rspinfo に保存し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S7F26(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TFPP_INFO rspinfo,              // S7F26 に含まれる FPP 情報です。 Vol-1 11.1 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意 味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

10．2．3．2　SendS7F25_wait()

S7F25 メッセージを送信し、引き続き応答メッセージの受信も行います。

【構文】

public int Send_wait(string fppid, ref TFPP_INFO rspinfo)

【引数】

fppid

FPPID です。

rspinfo

S7F26 応答メッセージに含まれるフォーマット付きプロセスプログラム情報保存するためのクラスのインスタンスです。

【戻り値】

返却値	意　味
0	正常に送信し、S7F26 を正常に受信した。
(-1)	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数に fppid が与えられない場合は、予め、プロパティ fppid を設定しておく必要があります。
実際の S7F25 送信処理までは、10.2.3.1 で説明したとおりですが、送信要求の後、S7F26 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

正常に受信できた場合は、rspinfo 内に FPP 情報を設定します。

送信または、受信エラーを検出した場合は、(-1)を返却します。(T3 ﾀｲﾑｱｳﾄも含む)
T3 タイムアウトが発生した場合、DSHDR2　HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. レシピ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S15F3,S15F4	class_SendS15F3	S15F3 送信　Recipe Name Space Action Request
		class_SendS15F4Response	S15F4 応答
2	S15F5,S15F6	class_SendS15F5	S15F5 送信　Recipe Namespace Rename Request
		class_SendS15F6Response	S15F6 応答
3	S15F7,S15F8	class_SendS15F7	S15F7 送信　Recipe Space Request
		-	(S15F8 はｴﾝｼﾞﾝが自動応答)
4	S15F9,S15F10	class_SendS15F9	S15F9 送信　Recipe Status Request
		-	(S15F10 はｴﾝｼﾞﾝが自動応答)
5	S15F13,S15F14	class_SendS15F13	S15F13 送信　Recipe Create Request
		class_SendS15F14Response	S15F14 応答
6	S15F17,S15F18	class_SendS15F17	S15F17 送信　Recipe Retrieve Request
		class_SendS15F18Response	S15F18 応答

[参照クラス]

	クラス名	参照場所
1	TRCP_INFO	エンジンクラス説明書-Vol-3 4.2 TRCP_INFO - レシピ情報保存クラス

11. 1 class_SendS15F3 クラス - Recipe Name Space Action Request

S15F3 メッセージを送信し、S15F4 応答メッセージを受信するためのクラスです。
レシピ名スペースのアクション要求を行います。

11. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F3()	空のｲﾝｽﾀﾝｽを生成します。

11. 1. 2 プロパティ

なし。

11. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F3()	S15F3 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS15F3_wait()	S15F3 ﾒｯｾｰｼﾞ を送信し、S15F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_ERR_INFO	エンジンクラス説明書-Vol-5 3. 18 TRCP_ERR_INFO - レシピ関連メッセージ応答情報クラス

11．1．3．1　SendS15F3()

S15F3 メッセージの送信要求をします。
応答メッセージ S15F4 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS15F3(string rcpid, int rmnscmd, class_CALLBACK.callback_S15F4 callback, UInt32 upara)

【引数】

rcpid

レシピ ID です。

rmnscmd

アクションコードです。

callback

S15F3 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	rcpid が登録されていなかった。 エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rcpid, rmnscmd から S15F3 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F4(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TRCP_ERR_INFO rsp_info,         // S15F4 に含まれる応答です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

11．1．3．2　SendS15F3_wait()

S15F3 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F3_wait(string rcpid, int rmnscmd, ref TRCP_ERR_INFO err_info)

【引数】

rcpid

　　レシピ ID です。

rmnscmd

　　アクションコードです。

rsp_info

　　S15F4 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

指定された rcpid, rmnscmd から S15F3 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F4 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報を rsp_info
に設定し渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. ２ class_SendS15F5 クラス - Recipe Name space Rename Request

S15F5 メッセージを送信し、S15F6 応答メッセージを受信するためのクラスです。
S15F5 は、レシピ名の新しい名前への変更要求を行います。

11. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F5()	空のｲﾝｽﾀﾝｽを生成します。

11. ２. ２ プロパティ

なし。

11. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F5()	S15F5 ﾒｯｾｰｼﾞを送信します。
2	public int SendS15F5_wait()	S15F5 ﾒｯｾｰｼﾞを送信し、S15F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_ERR_INFO	エンジンクラス説明書-Vol-5 3. 18 TRCP_ERR_INFO - レシピ関連メッセージ応答情報クラス

11．2．3．2　SendS15F5()

S15F5 メッセージの送信要求をします。

【構文】

```
public int SendS15F5(string rcpid, string new_rcpid,
                                    class_CALLBACK.callback_S15F6 callback, UInt32 upara))
```

【引数】

rcpid

　　現レシピ ID です。

new_rcp

　　新レシピ ID です。

callback

　　S15F5 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rcpid と new_rcpid から S15F5 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F6(
    int end_status,                     // 終了状態ｺｰﾄﾞ
    TRCP_ERR_INFO rsp_info,     // S15F6 に含まれる応答です。
    uint upara                          // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

11．２．３．３　Send_S15F5_wait()

S15F5 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F5_wait(string rcpid, string new_rcpid, ref TRCP_ERR_INFO rsp_info)

【引数】

rcpid

　　レシピ ID です。

new_rcp

　　新レシピ ID です。

rsp_info

　　S15F6 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

指定された rcpid と new_rcpid から S15F5 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報を rsp_info
に設定し渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. ３ class_SendS15F7 クラス

S15F7 メッセージを送信し、S15F8 応答メッセージを受信するためのクラスです。
S15F7 は、指定されたオブジェクトスペックの記憶装置内での記憶容量を取得するメッセージです。

11. ３. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F7()	空のｲﾝｽﾀﾝｽを生成します。

11. ３. ２ プロパティ

なし。

11. ３. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F7()	S15F7 ﾒｯｾｰｼﾞを送信します。
2	public int SendS15F7_wait()	S15F7 ﾒｯｾｰｼﾞを送信し、S15F8 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_ERR_INFO	エンジンクラス説明書-Vol-5 3. 18 TRCP_ERR_INFO - レシピ関連メッセージ応答情報クラス

11．３．３．１　SendS15F7()

S15F7 メッセージの送信要求をします。
応答メッセージ S15F8 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS15F7(string rcpid, class_CALLBACK.callback_S15F8 callback, UInt32 upara)

【引数】

rcpid

レシピ ID です。

callback

S15F7 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 objspec から S15F7 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
rsp_info の中の rmspace の値が記憶容量になります。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F8(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TRCP_ERR_INFO rsp_info,         // S15F8 に含まれる応答です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

11．３．３．２　SendS15F7_wait()

S15F7 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F7_wait(string rcpid, ref TRCP_S15F8_INFO rsp_info)

【引数】

rcpid

　　レシピ ID です。

rsp_info

　　S15F8 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

指定された rcpid から S15F7 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F8 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報を rsp_info
に設定し渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. ４ class_SendS15F9 クラス - Recipe Status Request

S15F9 メッセージを送信し、S15F10 応答メッセージを受信するためのクラスです。
S15F9 は、レシピの状態(state)とバージョン情報の取得を行います。

11. ４. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F9()	空のｲﾝｽﾀﾝｽを生成します。

11. ４. ２ プロパティ

なし。

11. ４. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F9()	S15F9 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS15F9_wait()	S15F9 ﾒｯｾｰｼﾞ を送信し、S15F10 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_S15F10_INFO	エンジンクラス説明書-Vol-5 3. 22 TRCP_S15F10_INFO - S15F9 メッセージ応答情報クラス

11．4．3．1　SendS15F9()

S15F9 メッセージの送信要求をします。
応答メッセージ S15F10 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS15F9(string rcpid, class_CALLBACK.callback_S15F10 callback, UInt32 upara)

【引数】

rcpid

レシピ ID です。

callback

S15F9 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rcpid から S15F9 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージ情報を中に保存した TRCP_S15F10_INFO のクラスのインスタンス rsp_info を引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F10(
    int end_status,                    // 終了状態ｺｰﾄﾞ
    TRCP_S15F10_INFO rsp_info,  // S15F10 に含まれる応答です。
    uint upara                         // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

11．４．３．２　SendS15F9_wait()

S15F9 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F9_wait(string rcpid, ref TRCP_S15F10_INFO rsp_info)

【引数】

rcpid

　　レシピ ID です。

rsp_info

　　S15F10 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

指定された rcpid から S15F9 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F10 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. ５ class_SendS15F13 クラス - Recipe Create Request

S15F13 メッセージを送信し、S15F14 応答メッセージを受信するためのクラスです。
S15F13 は、レシピ情報を送信するためのメッセージです。

11. ５. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F13()	空のｲﾝｽﾀﾝｽを生成します。

11. ５. ２ プロパティ

なし。

11. ５. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F13()	S15F13 ﾒｯｾｰｼﾞを送信します。
2	public int SendS15F13_wait()	S15F13 ﾒｯｾｰｼﾞを送信し、S15F14 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_INFO	エンジンクラス説明書-Vol-3 4.2 TRCP_INFO - レシピ情報保存クラス
2	TRCP_ERR_INFO	エンジンクラス説明書-Vol-5 3. 18 TRCP_ERR_INFO - レシピ関連メッセージ応答情報クラス

11．5．3．1　SendS15F13()

S15F13 メッセージの送信要求をします。
応答メッセージ S15F14 の情報は callback 関数の引数で渡されます。

【構文】

```
public int SendS15F13(string rcpid, bool updt_flag,
                      class_CALLBACK.callback_S15F14 callback, UInt32 upara)
```

【引数】

rcpid
　　レシピ ID です。
updt_flag
　　新規生成/更新の指定をします。(true=更新, false=新規生成)
callback
　　S15F13 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rcpid, updt_flag から S15F13 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
状態値とバージョン情報は、rsp_info の rcpstate と rcpver プロパティに設定されます。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F14(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TRCP_ERR_INFO rsp_info,         // S15F14 に含まれる応答です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

11．５．３．２　SendS15F13_wait()

S15F13 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F13_wait(string rcpid, bool updt_flag, ref TRCP_ERR_INFO rsp_info)

【引数】

rcpid

　　レシピ ID です。

updt_flag

　　新規生成/更新の指定をします。(true=更新, false=新規生成)

rsp_info

　　S15F14 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 rcpid, updt_flag から S15F13 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F14 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

11. ６ class_SendS15F17 クラス - Recipe Retrieve Request

S15F17 メッセージを送信し、S15F18 応答メッセージを受信するためのクラスです。
S15F17 は、レシピ検索要求を送信するためのメッセージです。

11. ６. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS15F17()	空のｲﾝｽﾀﾝｽを生成します。

11. ６. ２ プロパティ

なし。

11. ６. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS15F17()	S15F17 ﾒｯｾｰｼﾞを送信します。
2	public int SendS15F17_wait()	S15F17 ﾒｯｾｰｼﾞを送信し、S15F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCP_S15F18_INFO	エンジンクラス説明書-Vol-5 3. 24 TRCP_S15F18_INFO - S15F17 メッセージ応答情報クラス

11．6．3．1　SendS15F17()

S15F17 メッセージの送信要求をします。
応答メッセージ S15F18 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS15F17(string rcpid, int seccode, class_CALLBACK.callback_S15F18 callback, UInt32 upara)

【引数】

rcpid
　検索するレシピ ID です。
seccode
　検索するレシピセクションコードです。
callback
　S15F17 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。

【説明】

引数 rcpid と rcpseccode の情報から S15F17 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
rsp_info に検索結果情報が保存されます。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S15F18(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TRCP_S15F18_INFO rsp_info,      // S15F18 に含まれる応答です。  Vol-1 12.6 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-18	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

11．6．3．2　SendS15F17_wait()

S15F17 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS15F17_wait(string rcpid, int seccode, ref TRCP_S15F18_INFO rsp_info)

【引数】

rcpid

検索するレシピ ID です。

seccode

検索するレシピセクションコードです。

rsp_info

S15F18 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

指定された rcpid と rcpseccode の情報から S15F17 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S15F18 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. プロセス・ジョブ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S16F5, S16F6	class_SendS16F5	S16F5 送信　Process Job Command Request
		class_SendS16F6Response	S16F6 応答
2	S16F11, S16F12	class_SendS16F11	S16F11 送信　PrJobCreateEnh
		class_SendS16F12Response	S16F12 応答
3	S16F15, S16F16	class_SendS16F15	S16F15 送信　PrJobMultiCreate
		class_SendS16F16Response	S16F16 応答
4	S16F17, S16F18	class_SendS16F17	S16F17 送信　PrJobDeque
		class_SendS16F18Response	S16F18 応答
5	S16F19, S16F20	class_SendS16F19	S16F19 送信　PrGetAllJobs
		－	(S16F20 はｴﾝｼﾞﾝが自動応答)
6	S16F21, S16F22	class_SendS16F21	S16F21 送信　PrGetSpace
		－	(S16F22 はｴﾝｼﾞﾝが自動応答)

12. 1 class_SendS16F5 クラス - Process Job Command Request

S16F5 メッセージを送信し、S16F6 応答メッセージを受信するためのクラスです。
プロセスジョブのコマンド要求を行います。

12. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F5()	空のｲﾝｽﾀﾝｽを生成します。

12. 1. 2 プロパティ

なし。

12. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F5()	S16F5 ﾒｯｾｰｼﾞ を送信します。
2	public int Send_wait()	S16F5 ﾒｯｾｰｼﾞ を送信し、S16F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPRJ_CMD_INFO	エンジンクラス説明書-Vol-3 5.7 TPRJ_CMD_INFO - プロセスジョブ・コマンド情報クラス
2	TPRJ_ERR_INFO	エンジンクラス説明書-Vol-5 3. 26 TPRJ_ERR_INFO - プロセスジョブ関連メッセージ応答情報クラス

12. 1. 3. 1 SendS16F5()

S16F5 メッセージの送信要求をします。
応答メッセージ S16F6 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS16F5(TPRJ_CMD_INFO info, class_CALLBACK.callback_S16F6 callback, UInt32 upara)

【引数】

info

プロセスジョブコマンド情報が保存されている TPRJ_CMD_INFO のインスタンスです。

callback

S16F5 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S16F5 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F6(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TPRJ_ERR_INFO rsp_info,         // S16F6 に含まれる応答です。 Vol-1 13.5 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 1. 3. 2 SendS16F5_wait()

S16F5 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F5_wait(PRJ_CMD_INFO info, ref TPRJ_ERR_INFO rsp_info)

【引数】

info

プロセスジョブコマンド情報が保存されている TPRJ_CMD_INFO のインスタンスです。

rsp_info

S16F6 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 infos から S16F5 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. ２ class_SendS16F11 クラス - PrJobCreateEnh

S16F11 メッセージを送信し、S16F12 応答メッセージを受信するためのクラスです。
プロセスジョブ情報の送信を行います。

12. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F11()	空のｲﾝｽﾀﾝｽを生成します。

12. ２. ２ プロパティ

なし。

12. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F11()	S16F11 ﾒｯｾｰｼﾞを送信します。
2	public int Send S16F11_wait()	S16F11 ﾒｯｾｰｼﾞを送信し、S16F12 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPRJ_INFO	エンジンクラス説明書-Vol-3 5.2 TPRJ_LIST - プロセスジョブ情報保存クラス
2	TPRJ_ERR_INFO	エンジンクラス説明書-Vol-5 3. 26 TPRJ_ERR_INFO - プロセスジョブ関連メッセージ応答情報クラス

12. 2. 3. 1 SendS16F11()

S16F11 メッセージの送信要求をします。
応答メッセージ S16F12 の情報は callback 関数の引数で渡されます。

【構文】

```
public int SendS16F11(string prjid, class_CALLBACK.callback_S16F12 callback, UInt32 upara)
```

【引数】

prjid
　　プロセスジョブ ID です。
callback
　　S16F11 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	要求が受け入れられなかった。または prjid が管理外であった。 prjid が登録されていなかった。

【説明】

引数 prjid から S16F11 メッセージを生成し、それを相手装置へ送信するように要求します。

プロセスジョブ ID が引数に指定されていた場合にはそのプロセスジョブ情報をエンジンの管理から prj_class プロパティに取得した上で送信処理をします。
また、DshPrj クラスのインスタンスが引数に与えられた場合には、それを prj_class プロパティにコピーした上で送信処理を行います。

要求がエンジンによって受け入れられたときは、0 を、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージ情報を保存した rsp_info 引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F12(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TPRJ_ERR_INFO rsp_info,         // S16F12 に含まれる応答です。 Vol-1 13.5 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 2. 3. 2 SendS16F11_wait()

S16F11 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F11_wait(string prjid, ref TPRJ_ERR_INFO rsp_info)

【引数】

prjid

プロセスジョブ ID です。

rsp_info

S16F12 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	prjid が登録されていなかった。 エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 prjid から S16F11 メッセージを生成し、それを相手装置へ送信するように要求します。

プロセスジョブ ID が引数に指定されていた場合にはそのプロセスジョブ情報をエンジンの管理から prj_class プロパティに取得した上で送信処理をします。
また、DshPrj クラスのインスタンスが引数に与えられた場合には、それを prj_class プロパティにコピーした上で送信処理を行います。

送信要求の後、S16F12 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. ３ class_SendS16F15 クラス - PrJobMultiCreate

S16F15 メッセージを送信し、S16F16 応答メッセージを受信するためのクラスです。
複数個のプロセスジョブ ID 分のプロセスジョブ情報の送信を行います。

12. ３. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F15()	空のｲﾝｽﾀﾝｽを生成します。

12. ３. ２ プロパティ

なし。

12. ３. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F15()	S16F15 ﾒｯｾｰｼﾞを送信します。
2	public int SendS16F15_wait()	S16F15 ﾒｯｾｰｼﾞを送信し、S16F16 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPRJ_LIST	エンジンクラス説明書-Vol-3 5.3 TPRJ_LIST - プロセスジョブ情報保存リストクラス
2	TPRJ_ERR_INFO	エンジンクラス説明書-Vol-5 3. 26 TPRJ_ERR_INFO - プロセスジョブ関連メッセージ応答情報クラス

12. 3. 3. 1 SendS16F15()

S16F15 メッセージの送信要求をします。
応答メッセージ S16F16 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS16F15(TPRJ_LIST list, class_CALLBACK.callback_S16F16 callback, UInt32 upara)

【引数】

list

複数のプロセスジョブ情報が保存されている TPRJ_LISU クラスのインスタンスです。

callback

S16F15 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	id が管理外であった。 エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list に含まれる 1 個以上のプロセスジョブ ID のプロセスジョブ情報をエンジンの管理情報から取得し、S16F15 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F16(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TMPRJ_ERR_INFO rsp_info,        // S16F16 に含まれる応答です。 Vol-1 13.6 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 3. 3. 2 Send16F15_wait()

S16F15 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F15_wait(TPRJ_LIST list, ref TMPRJ_ERR_INFO rsp_info)

【引数】

list

複数のプロセスジョブ情報が保存されている TPRJ_LIST クラスのインスタンスです。

rsp_info

S16F16 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 list に含まれる 1 個以上のプロセスジョブ ID のプロセスジョブ情報をエンジンの管理情報から取得し、S16F15 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F16 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. ４ class_SendS16F17 クラス - PrJobDeque

S16F17 メッセージを送信し、S16F18 応答メッセージを受信するためのクラスです。
複数個のプロセスジョブ ID 分のプロセスジョブ削除のための送信を行います。

12. ４. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F17()	空のｲﾝｽﾀﾝｽを生成します。

12. ４. ２ プロパティ

なし。

12. ４. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F17()	S16F17 ﾒｯｾｰｼﾞを送信します。
2	public int SendS16F17_wait()	S16F17 ﾒｯｾｰｼﾞを送信し、S16F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPRJ_DEQ_INFO	エンジンクラス説明書-Vol-3 5.4 TPRJ_DEQ_INFO - プロセスジョブ削除 ID リストクラス
2	TMPRJ_ERR_INFO	エンジンクラス説明書-Vol-5 3. 26 TPRJ_ERR_INFO - プロセスジョブ関連メッセージ応答情報クラス

12. 4. 3. 1 SendS16F17()

S16F17 メッセージの送信要求をします。
応答メッセージ S16F18 の情報は callback 関数の引数で渡されます。

【構文】

```
public int SendS16F17(TPRJ_DEQ_INFO list, class_CALLBACK.callback_S16F18 callback, UInt32 upara)
```

【引数】

list
　　削除したいプロセスジョブの配列リストです。
callback
　　S16F17 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 list 配列に含まれる 1 個以上のプロセスジョブ ID から S16F17 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F18(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TMPRJ_ERR_INFO rsp_info,        // S16F18 に含まれる応答です。 Vol-1 13.6 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 4. 3. 2 SendS16F17_wait()

S16F17 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F17_wait(TPRJ_DEQ_INFO list, ref TMPRJ_ERR_INFO err_info)

【引数】

list

削除したいプロセスジョブの配列リストです。

rsp_info

S16F18 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 list 配列に含まれる 1 個以上のプロセスジョブ ID から S16F17 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F18 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. ５ class_SendS16F19 クラス - PrGetAllJobs

S16F19 メッセージを送信し、S16S20 応答メッセージを受信するためのクラスです。
相手装置から現存するプロセスジョブ ID と状態を取得します。

12. ５. １ コンストラクタ

	名前	説明
1	public class_SendS16F19()	空のｲﾝｽﾀﾝｽを生成します。

12. ５. ２ プロパティ

なし。

12. ５. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F19()	S16F19 ﾒｯｾｰｼﾞを送信します。
2	public int SendS16F19_wait()	S16F19 ﾒｯｾｰｼﾞを送信し、S16F20 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPRJ_STATE_LIST	エンジンクラス説明書-Vol-3 5.5 TPRJ_STATE_LIST - プロセスジョブ状態情報リストクラス

12. 5. 3. 1 SendS16F19()

S16F19 メッセージの送信要求をします。
応答メッセージ S16F20 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS16F19(class_CALLBACK.callback_S16F19 callback, uint upara)

【引数】

callback

S16F19 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

S16F19 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F20(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TPRJ_STATE_LIST rsp_info,       // S16S20 に含まれる応答です。 Vol-1 13.7 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 5. 3. 2 SendS16F19_wait()

S16F19 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F19_wait(ref TPRJ_STATE_LIST rsp_info)

【引数】

rsp_info

S16F20 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

S16F19 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F20 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる応答情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

12. ６ class_SendS16F21 クラス - PrGetSpace

S16F21 メッセージを送信し、S16S20 応答メッセージを受信するためのクラスです。
相手装置からプロセスジョブのための生成可能スペース数を取得します。

12. ６. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F21()	空のｲﾝｽﾀﾝｽを生成します。

12. ６. ２ プロパティ

なし。

12. ６. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F21()	S16F21 ﾒｯｾｰｼﾞを送信します。
2	public int Send_wait()	S16F21 ﾒｯｾｰｼﾞを送信し、S16F22 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

12. 6. 3. 1 SendS16F21()

S16F21 メッセージの送信要求をします。
応答メッセージ S16F22 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS16F21(class_CALLBACK.callback_S16F21 callback, uint upara)

【引数】

callback

S16F21 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

S16F21 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F22(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int prjspace,                   // S16S20 に含まれるﾌﾟﾛｾｽｼﾞｮﾌﾞの生成可能ｽﾍﾟｰｽ数です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

12. 6. 3. 2 SendS16F21_wait()

S16F21 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F21_wait(ref int prjspace)

【引数】

prjspace

S16S20 応答メッセージに含まれるスペース数を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど）

【説明】

S16F21 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F22 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージに含まれる PRJ スペース数が prjspace に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

13. コントロール・ジョブ関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S14F9,S14F10	class_SendS14F9	S14F9 送信　Create Object Request
2	S14F11,S14F12	class_SendS14F11	S14F11 送信　Delete Object Request
		class_SendS14F12Response	S14F12 応答
3	S16F27,S16F28	class_SendS16F27	S16F27 送信　Control Job Command Request
		class_SendS16F28Response	S16F28 応答

13. 1 class_SendS14F9 クラス - Create Object Request

S14F9 メッセージを送信し、S14F10 応答メッセージを受信するためのクラスです。
コントロールジョブ生成情報の送信を行います。

13. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS14F9()	空のｲﾝｽﾀﾝｽを生成します。

13. 1. 2 プロパティ

なし。

13. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS14F9()	S14F9 ﾒｯｾｰｼﾞを送信します。
2	public int SendS14F9_wait()	S14F9 ﾒｯｾｰｼﾞを送信し、S14F10 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TCJ_INFO	エンジンクラス説明書-Vol-3 6.2 TCJ_INFO - コントロール・ジョブ情報保存クラス
2	TOBJ_S14_ERR_INFO	エンジンクラス説明書-Vol-5 3. 17 TOBJ_S14_ERR_INFO - コントロールジョブメッセージ応答情報クラス

13．1．3．1　SendS14F9()-

S14F9 メッセージの送信要求をします。
応答メッセージ S14F10 の情報は callback 関数の引数で渡されます。

【構文】

public int SendS14F9(string cjid, class_CALLBACK.callback_S14F10 callback, UInt32 upara)

【引数】

cjid

コントロールジョブ ID を指定します。

callback

S14F9 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	cjid が登録されていなかった。 エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 cjid から S14F9 メッセージを生成し、それを相手装置へ送信するように要求します。

引数にコントロールジョブ ID が指定されていた場合にはそのコントロールジョブ情報をエンジンの管理から cj_class プロパティに取得した上で送信処理をします。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S14F10(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TOBJ_S14_ERR_INFO rsp_info,     // S14F10 に含まれる応答です。 Vol-1 14.14 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

13. 1. 3. 2 Send S14F9_wait()

S14F9 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS14F9_wait(string cjid, ref TOBJ_S14_ERR_INFO rsp_info)

【引数】

cjid

コントロールジョブ ID を指定します。エンジンから情報を cj_class に取得します。

rsp_info

S14F10 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 cjid から S14F9 メッセージを生成し、それを相手装置へ送信するように要求します。

引数にコントロールジョブ ID が指定されていた場合にはそのコントロールジョブ情報をエンジンの管理から cj_class プロパティに取得した上で送信処理をします。

送信要求の後、S14F10 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

13. ２ class_SendS14F11 クラス - Delete Object Request

S14F11 メッセージを送信し、S14F12 応答メッセージを受信するためのクラスです。
コントロールジョブ削除情報の送信を行います。

13. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS14F11()	空のｲﾝｽﾀﾝｽを生成します。

13. ２. ２ プロパティ

なし。

13. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS14F11()	S14F11 ﾒｯｾｰｼﾞを送信します。
2	public int SendS14F11_wait()	S14F11 ﾒｯｾｰｼﾞを送信し、S14F12 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TCJ_INFO	エンジンクラス説明書-Vol-3 6.2 TCJ_INFO - コントロール・ジョブ情報保存クラス
2	TOBJ_S14_ERR_INFO	エンジンクラス説明書-Vol-5 3. 17 TOBJ_S14_ERR_INFO - コントロールジョブメッセージ応答情報クラス

13. 2. 3. 1 SendS14F11()

S14F11 メッセージの送信要求をします。
応答メッセージ S14F12 の情報は callback 関数の引数で渡されます。

【構文】

```
public int SendS14F11(string cjid, class_CALLBACK.callback_S14F12 callback, UInt32 upara)
```

【引数】

cjid

コントロールジョブ ID を指定します。

callback

S14F11 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 cjid から S14F11 メッセージを生成し、それを相手装置へ送信するように要求します。

コントロールジョブ ID が引数に指定されていた場合にはそのコントロールジョブ情報をエンジンの管理から cj_class プロパティに取得した上で送信処理をします。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S14F12(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TOBJ_S14_ERR_INFO rsp_info,     // S14F12 に含まれる応答です。 Vol-1 14.14 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

13．2．3．2　SendS14F11_wait()

S14F11 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS14F11_wait(string cjid, ref TOBJ_S14_ERR_INFO rsp_info)

【引数】

cjid

コントロールジョブ ID を指定します。エンジンから情報を cj_class に取得します。

rsp_info

S14F12 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 cjid から S14F11 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S14F12 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

13. ３ class_SendS16F27 クラス - Control Job Command Request

S16F27 メッセージを送信し、S16F28 応答メッセージを受信するためのクラスです。
コントロールジョブのコマンド要求を行います。

本クラスは TCJ_CMD_INFO クラスを使用します。

13. ３．１ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS16F27()	空のｲﾝｽﾀﾝｽを生成します。

13. ３．２ プロパティ

なし。

13. ３．３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS16F27()	S16F27 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS16F27_wait()	S16F27 ﾒｯｾｰｼﾞ を送信し、S16F28 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TCJ_CMD_INFO	エンジンクラス説明書-Vol-3 6.11 TCJ_CMD_INFO - コントロールジョブ・コマンド情報クラス
2	TCJ_CMD_ERR_INFO	エンジンクラス説明書-Vol-5 3. 28 TCJ_CMD_ERR_INFO - コントロールジョブコマンドメッセージ応答情報クラス

13. 3. 3. 1 SendS16F27()

S16F27 メッセージの送信要求をします。
応答メッセージ S16F28 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS16F27(TCJ_CMD_INFO info, class_CALLBACK.callback_S16F28 callback, UInt32 upara)

【引数】

info

CJ コマンド情報です。

callback

S16F27 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info に設定された CJ コマンド情報から、S16F27 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S16F28(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TCJ_CMD_ERR_INFO rsp_info,      // S16F28 に含まれる応答です。 Vol-1 14.15 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

13．３．３．２　SendS16F27_wait()

S16F27 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS16F27_wait(TCJ_CMD_INFO info, ref class_TCJ_CMD_ERR_INFO rsp_info)

【引数】

info

CJ コマンド情報です。

rsp_info

S16F28 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど）

【説明】

引数 info に設定された CJ コマンド情報から、S16F27 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S16F28 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

14. スプール関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S2F43,44	class_SendS2F43	S2F43 送信　Reset Spooling Stream and Function
		class_SendS2F44Response	S2F44 応答
2	S6F23,S6F24	class_SendS6F23	S6F23 送信　Request Spooled Data
		-	(S6F24 はｴﾝｼﾞﾝが自動応答)

14. 1 class_SendS2F43 クラス - Reset Spooling Stream and Function

S2F43 メッセージを送信し、S2F44 応答メッセージを受信するためのクラスです。
スプール情報の送信を行います。

14. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F43()	空のｲﾝｽﾀﾝｽを生成します。

14. 1. 2 プロパティ

なし。

14. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F43()	S2F43 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F43_wait()	S2F43 ﾒｯｾｰｼﾞを送信し、S2F44 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TSPOOL_INFO	エンジンクラス説明書-Vol-2 3. 21 TSPOOL_INFO - スプール情報保存クラス
2	TSPOOL_ERR_INFO	エンジンクラス説明書-Vol-5 3. 29 T TSPOOL_ERR_INFO - S2F44 メッセージ情報クラス

14．1．3．1　SendS2F43()

S2F43 メッセージの送信要求をします。
応答メッセージ S2F44 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F43(TSPOOL_INFO info, class_CALLBACK.callback_S2F44 callback, UInt32 upara)

【引数】

info

スプール情報が保存されています。

callback

S2F43 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info 情報から S2F43 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F44(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TSPOOL_ERR_INFO rsp_info,       // S2F44 に含まれる応答です。 Vol-5  3.29 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-13	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

14．1．3．2　SendS2F43_wait()

S2F43 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F43_wait(TSPOOL_INFO info, ef TSPOOL_ERR_INFO rsp_info)

【引数】

info

　　スプール情報が保存されています。

rsp_info

　　S2F44 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 info 情報から S2F43 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F44 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に設定され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

14. ２ class_SendS6F23 クラス - Request Spooled Data

S6F23 メッセージを送信し、S6F24 応答メッセージを受信するためのクラスです。
スプールデータの転送または削除要求の送信を行います。

14. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS6F23()	空のｲﾝｽﾀﾝｽを生成します。

14. ２. ２ プロパティ

なし。

14. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS6F23()	S6F23 ﾒｯｾｰｼﾞを送信します。
2	public int SendS6F23_wait()	S2F23 ﾒｯｾｰｼﾞを送信し、S2F24 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

14. 2. 3. 1 SendS6F23()

S6F23 メッセージの送信要求をします。
応答メッセージ S6F24 の情報は callback 関数の引数に渡されます。

【構文】

```
public int SendS6F23(int rsdc, class_CALLBACK.callback_S6F23 callback, uint upara)
```

【引数】

rsdc

スプールデータの転送 / 削除を指定します。(0=転送、1=削除)

callback

S6F23 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 rsdc を含めた S6F23 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ACK を rsda に保存し、それを引数にして、callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S6F24(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int rsda,                       // S6F24 に含まれる応答 ack です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-13	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

14．２．３．２　SendS6F23_wait()

S6F23 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS6F23_wait(int rsdc, ref int rsda)

【引数】

rsdc

スプールデータの転送、削除を指定します。(0=転送、1=削除)

rsda

S6F24 応答メッセージに含まれる ack 情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 rsdc を含めた S6F23 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S6F24 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージの ACK が rsda に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. ホストコマンド、キャリアアクション関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S2F41, 42	class_SendS2F41	S2F41 送信　Host Command Send
		class_SendS2F42Response	S2F42 応答
2	S2F49, 50	class_SendS2F49	S2F49 送信　Enhanced Remote Command
		class_SendS2F50Response	S2F50 応答
3	S3F17, 18	class_SendS3F17	S3F17 送信　Carrier Action Request
		class_SendS3F18Response	S3F18 応答
4	S3F23, 24	class_SendS3F23	S3F23 送信　Port Group Action Request
		class_SendS3F24Response	S3F24 応答
5	S3F25, 26	class_SendS3F25	S3F25 送信　Port Action Request
		class_SendS3F26Response	S3F26 応答
6	S3F27, 28	class_SendS3F27	S3F27 送信　Change Access
		class_SendS3F28Response	S3F28 応答

15. 1 class_SendS2F41 クラス - Host Command Send

S2F41 メッセージを送信し、S2F42 応答メッセージを受信するためのクラスです。
ホストリモートコマンド情報の送信を行います。

15. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F41()	空のｲﾝｽﾀﾝｽを生成します。

15. 1. 2 プロパティ

なし。

15. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F41()	S2F41 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F41_wait()	S2F41 ﾒｯｾｰｼﾞを送信し、S2F42 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TRCMD_INFO	エンジンクラス説明書-Vol-5 3.5 ホストコマンド情報保存クラス
2	TRCMD_ERR_INFO	エンジンクラス説明書-Vol-5 3. 30 TRCMD_ERR_INFO - S2F42 メッセージ情報クラス

15．1．3．1　SendS2F41()

S2F41 メッセージの送信要求をします。
応答メッセージ S2F42 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F41(TRCMD_INFO info, class_CALLBACK.callback_S2F42 callback, UInt32 upara)

【引数】

info

リモートコマンド情報です。

callback

S2F41 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S2F41 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F42(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TRCMD_ERR_INFO rsp_info,        // S2F42 に含まれる応答です。 Vol-1 16.10 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15. 1. 3. 2 SendS2F41_wait()

S2F41 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F41_wait(TRCMD_INFO info, ref TRCMD_ERR_INFO rsp_info)

【引数】

info

リモートコマンド情報です。

rsp_info

S2F42 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 info から S2F41 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F42 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. 2 class_SendS2F49 クラス - Enhanced Remote Command

S2F49 メッセージを送信し、S2F50 応答メッセージを受信するためのクラスです。
ホストリモートコマンド情報の送信を行います。

15. 2. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F49()	空のｲﾝｽﾀﾝｽを生成します。

15. 2. 2 プロパティ

なし。

15. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F49()	S2F49 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F49_wait()	S2F49 ﾒｯｾｰｼﾞを送信し、S2F50 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TERCMD_INFO	エンジンクラス説明書-Vol-5 3. 6 TERCMD_INFO - 拡張リモートコマンド情報クラス
2	TRCMD_ERR_INFO	エンジンクラス説明書-Vol-5 3. 30 TRCMD_ERR_INFO - S2F42 メッセージ情報クラス

15．2．3．1　SendS2F49()

S2F49 メッセージの送信要求をします。
応答メッセージ S2F50 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F49(TERCMD_INFO info, class_CALLBACK.callback_S2F50 callback, UInt32 upara)

【引数】

info

拡張リモートコマンド情報です。

callback

S2F49 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S2F49 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F50(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TERCMD_ERR_INFO rsp_info,       // S2F50 に含まれる応答です。 Vol-1 16.10 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．２．３．２　SendS2F49_wait()

S2F49 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS2F49_wait(TERCMD_INFO info, ref TRCMD_ERR_INFO rsp_info)

【引数】

info

　　拡張リモートコマンド情報です。

rsp_info

　　S2F50 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど） またはエンコードに失敗した。

【説明】

引数 info から S2F49 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F50 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. 3 class_SendS3F17 クラス - Carrier Action Request

S3F17 メッセージを送信し、S3F18 応答メッセージを受信するためのクラスです。
キャリアアクション情報の送信を行います。

15. 3. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS3F17()	空のｲﾝｽﾀﾝｽを生成します。

15. 3. 2 プロパティ

なし。

15. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS3F17()	S3F17 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS3F17_wait()	S3F17 ﾒｯｾｰｼﾞ を送信し、S3F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TCACT_INFO	エンジンクラス説明書-Vol-5 3. 10 TCACT_INFO - キャリアアクション情報クラス
2	TCACT_ERR_INFO	エンジンクラス説明書-Vol-5 3. 11 TCACT_ERR_INFO -ャリアアクション、ポートアクション応答情報クラス

15. 3. 3. 1 SendS3F17()

S3F17 メッセージの送信要求をします。
応答メッセージ S3F18 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS3F17(TCACT_INFO info, class_CALLBACK.callback_S3F18 callback, UInt32 upara)

【引数】

info

キャリアアクション情報です。
(TCACT_INFO クラスは、エンジン・クラス説明書　vol-5 の 3.10 を参照ください。)

callback

S3F17 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S3F17 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S3F18(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TCACT_ERR_INFO rsp_info,        // S3F18 に含まれる応答です。 Vol-1 16.12 参照
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．３．３．２　SendS3F17_wait()

S3F17 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS3F17_wait(TCACT_INFO info, ref TCACT_ERR_INFO rsp_info)

【引数】

rsp_info

S3F18 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 info から S3F17 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S3F18 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. 4 class_SendS3F23 クラス - Port Group Action Request

S3F23 メッセージを送信し、S3F24 応答メッセージを受信するためのクラスです。
ポートグループアクション情報の送信を行います。

15. 4. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS3F23()	空のｲﾝｽﾀﾝｽを生成します。

15. 4. 2 プロパティ

なし。

15. 4. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS3F23()	S3F23 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS3F23_wait()	S3F23 ﾒｯｾｰｼﾞ を送信し、S3F24 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPORTG_INFO	エンジンクラス説明書-Vol-5 3. 12 TPORTG_INFO - ポートグループ情報クラス
2	TCACT_ERR_INFO	エンジンクラス説明書-Vol-5 3. 11 TCACT_ERR_INFO -ャリアアクション、ポートアクション応答情報クラス

15．4．3．1　SendS3F23()

S3F23 メッセージの送信要求をします。
応答メッセージ S3F24 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS3F23(TPORTG_INFO info, class_CALLBACK.callback_S3F24 callback, UInt32 upara)

【引数】

info

ポートグループアクション情報です。

callback

S3F23 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S3F23 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S3F24(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TCACT_ERR_INFO rsp_info,        // S3F24 に含まれる応答です。 Vol-1 16.13 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．４．３．２　SendS3F23_wait()

S3F23 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS3F23_wait(TPORTG_INFO info, ref TCACT_ERR_INFO rsp_info)

【引数】

info

　　ポートグループアクション情報です

rsp_info

　　S3F24 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 info から S3F23 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S3F24 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. ５ class_SendS3F25 クラス - Port Action Request

S3F25 メッセージを送信し、S3F26 応答メッセージを受信するためのクラスです。
ポートアクション情報の送信を行います。

15. ５. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS3F25()	空のｲﾝｽﾀﾝｽを生成します。

15. ５. ２ プロパティ

なし。

15. ５. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS3F25()	S3F25 ﾒｯｾｰｼﾞ を送信します。
2	public int SendS3F25_wait()	S3F25 ﾒｯｾｰｼﾞ を送信し、S3F26 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TPORT_INFO	エンジンクラス説明書-Vol-5 3. 13 TPORT_INFO - ポート情報クラス
2	TCACT_ERR_INFO	エンジンクラス説明書-Vol-5 3. 11 TCACT_ERR_INFO -ャリアアクション、ポートアクション応答情報クラス

15. 5. 3. 1 SendS3F25()

S3F25 メッセージの送信要求をします。
応答メッセージ S3F26 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS3F25(TPORT_INFO info, class_CALLBACK.callback_S3F26 callback, UInt32 upara)

【引数】

info

ポートアクション情報です。

callback

S3F25 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S3F25 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S3F26(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TCACT_ERR_INFO rsp_info,        // S3F26 に含まれる応答です。 Vol-1 16.13 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．５．３．２　SendS3F25_wait()

S3F25 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS3F25_wait(TPORT_INFO info, ref TCACT_ERR_INFO rsp_info)

【引数】

info

ポートアクション情報です。

rsp_info

S3F26 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 info から S3F25 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S3F26 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

15. ６ class_SendS3F27 クラス - Change Access

S3F27 メッセージを送信し、S3F28 応答メッセージを受信するためのクラスです。
ポートアクセス変更情報の送信を行います。

15. ６. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS3F27()	空のｲﾝｽﾀﾝｽを生成します。

15. ６. ２ プロパティ

なし。

15. ６. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS3F27()	S3F27 ﾒｯｾｰｼﾞを送信します。
2	public int SendS3F27_wait()	S3F27 ﾒｯｾｰｼﾞを送信し、S3F28 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TACCESS_INFO	エンジンクラス説明書-Vol-5 3. 14 TACCESS_INFO - ポートアクセス情報クラス
2	TACCESS_ERR_INFO	エンジンクラス説明書-Vol-5 3. 15 TACCESS_ERR_INFO - ポートアクセスエラー情報クラス

15．6．3．1　SendS3F27()

S3F27 メッセージの送信要求をします。
応答メッセージ S3F28 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS3F27(TACCESS_INFO info, class_CALLBACK.callback_S3F28 callback, UInt32 upara)

【引数】

info

ポートアクセス変更情報です。

callback

S3F27 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info から S3F27 メッセージを生成し、それを相手装置へ送信するように要求します。

要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージをデコードし、応答情報を作成し、それを引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S3F28(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    TACCESS_ERR_INFO rsp_info,      // S3F28 に含まれる応答です。 Vol-1 16.15 参照
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

15．6．3．2　SendS3F27_wait()

S3F27 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(TACCESS_INFO info, ref TACCESS_ERR_INFO rsp_info)

【引数】

info

ポートアクセス変更情報です。

rsp_info

S3F28 応答メッセージに含まれる情報を保存します。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

引数 info から S3F27 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S3F28 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答メッセージ情報が rsp_info に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

16. 端末表示関連メッセージの送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S10F1,S10F2	class_SendS10F1	S10F1 送信　Terminal Request
		class_SendS10F2Response	S10F2 応答
2	S10F3,S10F4	class_SendS10F3	S10F3 送信　Treminal Display, Single
		class_SendS10F4Response	S10F4 応答
3	S10F5,S10F6	class_SendS10F5	S10F5 送信　Terminal Display, Multi-Block
		class_SendS10F6Response	S10F6 応答

16. 1 class_SendS10F1 クラス - S10F1 送信 Terminal Request

S10F1 メッセージを送信し、S10F2 応答メッセージを受信するためのクラスです。

16. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS10F1()	ｲﾝｽﾀﾝｽを生成します。

16. 1. 2 プロパティ

なし。

16. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS10F1()	S10F1 ﾒｯｾｰｼﾞを送信します。
2	public int SendS10F1_wait()	S10F1 ﾒｯｾｰｼﾞを送信し、S10F2 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。S10F1

16. 1. 3. 1 SendS10F1()

S10F1 メッセージの送信要求をします。
応答メッセージ S10F2 の情報は callback 関数の引数に渡されます。

【構文】

```
public int SendS10F1( int tid, string text, class_CALLBACK.callback_S10F2 callback, UInt32 upara)
```

【引数】

tid
　端末 ID です。
text
　表示テキストです。
callback
　S10F1 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 tid、text から S10F1 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ackc10 を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S10F2(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc10,                     // S10F2 に含まれる ackc10 です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

16. 1. 3. 2 SendS10F1_wait()

S10F1 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

```
public int SendS10F1_wait(int tid, string text, ref int ack10)
```

【引数】

tid

端末 ID です。

text

表示テキストです。

ackc10

S10F2 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 tid、text から S10F1 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S10F2 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が ackc10 に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

16. 2 class_SendS10F3 クラス - Treminal Display, Single

S10F3 メッセージを送信し、S10F4 応答メッセージを受信するためのクラスです。

16. 2. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS10F3()	ｲﾝｽﾀﾝｽを生成します。

16. 2. 2 プロパティ

なし。

16. 2. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS10F3()	S10F3 ﾒｯｾｰｼﾞを送信します。
2	public int SendS10F3_wait()	S10F3 ﾒｯｾｰｼﾞを送信し、S10F4 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。S10F3

16. 2. 3. 1 SendS10F3()

S10F3 メッセージの送信要求をします。
応答メッセージ S10F4 の情報は callback 関数の引数に渡されます。

【構文】

```
public int SendS10F3( int tid, string text, class_CALLBACK.callback_S10F4 callback, UInt32 upara)
```

【引数】

tid
　端末 ID です。
text
　表示テキストです。
callback
　S10F3 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。
upara
　ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
　要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 tid、text から S10F3 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ackc10 を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S10F4(
    int end_status,                    // 終了状態ｺｰﾄﾞ
    int ackc10,                        // S10F4 に含まれる ackc10 です。
    uint upara                         // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

16. 2. 3. 2 SendS10F3_wait()

S10F3 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS10F3_wait(int tid, string text, ref int ack10)

【引数】

tid

端末 ID です。

text

表示テキストです。

ackc10

S10F4 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 tid、text から S10F3 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S10F4 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が ackc10 に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

16. 3 class_SendS10F5 クラス - Terminal Display, Multi-Block

S10F5 メッセージを送信し、S10F6 応答メッセージを受信するためのクラスです。

16. 3. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS10F5()	ｲﾝｽﾀﾝｽを生成します。

16. 3. 2 プロパティ

なし。

16. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS10F5()	S10F5 ﾒｯｾｰｼﾞを送信します。
2	public int Send_wait()	S10F5 ﾒｯｾｰｼﾞを送信し、S10F6 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

[参照クラス]

	クラス名	参照場所
1	TTERM_MULTI_INFO	エンジンクラス説明書-Vol-5 3.32 TTERM_MULTI_INFO - 端末表示テキストリスト情報クラス

16. 3. 3. 1 SendS10F5()

S10F5 メッセージの送信要求をします。
応答メッセージ S10F6 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS10F5(TTERM_MULTI_INFO info, class_CALLBACK.callback_S10F6 callback, UInt32 upara)

【引数】

info

端末 ID, 複数テキストの表示情報です。

callback

S10F5 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 info に保存されている端末表示情報から S10F5 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの ackc10 を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S10F6(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int ackc10,                     // S10F6 に含まれる ackc10 です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

16. 3. 3. 2 SendS10F5_wait()

S10F5 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS10F5_wait(TTERM_MULTI_INFO info, ref int ack10)

【引数】

info

端末 ID, 複数テキストの表示情報です。

ackc10

S10F6 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど)

【説明】

引数 info に保存されている端末表示情報から S10F5 メッセージを生成し､それを相手装置へ送信するように要求します。

送信要求の後、S10F6 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が ackc10 に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合､DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

17. オンライン／オフライン、日付時刻設定メッセージ送受信

以下のメッセージとクラスがあります。

	メッセージ	クラス名	機能概略
1	S1F15,16	class_SendS1F15	S1F15 送信　Request OFF-LINE
		class_SendS1F16Response	S1F16 応答
2	S1F17,18	class_SendS1F17	S1F17 送信　Request ON-LINE
		class_SendS1F18Response	S1F18 応答
3	S2F31,32	class_SendS2F31	S2F31 送信　Date and Time Set Requst
		－	(S2F32 はｴﾝｼﾞﾝが自動応答)

17. 1 class_SendS1F15 クラス - Request OFF-LINE

S1F15 メッセージを送信し、S1F16 応答メッセージを受信するためのクラスです。
S1F15 はオフライン要求のメッセージです。

17. 1. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS1F15()	ｲﾝｽﾀﾝｽを生成します。

17. 1. 2 プロパティ

なし。

17. 1. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS1F15()	S1F15 ﾒｯｾｰｼﾞを送信します。
2	public int SendS1F15_wait()	S1F15 ﾒｯｾｰｼﾞを送信し、S1F16 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

17．1．3．1　SendS1F15()

S1F15 メッセージの送信要求をします。
応答メッセージ S1F16 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS1F15(class_CALLBACK.callback_S1F16 callback, UInt32 upara)

【引数】

callback

S1F15 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

S1F15 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの oflack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S1F16(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int oflack,                     // S1F16 に含まれる oflack です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

17．1．3．2　SendS1F15_wait()

S1F15 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(ref int oflack)

【引数】

oflack

S1F16 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

S1F15 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S1F16 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が oflack に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

17. ２ class_SendS1F17 クラス - Request ON-LINE

S1F17 メッセージを送信し、S1F18 応答メッセージを受信するためのクラスです。
S1F17 はオンライン要求のメッセージです。

17. ２. １ コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS1F17()	ｲﾝｽﾀﾝｽを生成します。

17. ２. ２ プロパティ

なし。

17. ２. ３ メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS1F17()	S1F17 ﾒｯｾｰｼﾞを送信します。
2	public int SendS1F17_wait()	S1F17 ﾒｯｾｰｼﾞを送信し、S1F18 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

17．2．3．1　SendS1F17()

S1F17 メッセージの送信要求をします。
応答メッセージ S1F18 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS1F17(class_CALLBACK.callback_S1F18 callback, UInt32 upara)

【引数】

callback

S1F17 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

S1F17 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの onlack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S1F18(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int onlack,                     // S1F18 に含まれる onlack です。
    uint upara                      // ﾕｰｻﾞﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

17．２．３．２　SendS1F17_wait()

S1F17 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int SendS1F17_wait(ref int onlack)

【引数】

onlack

S1F18 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。（送信ｴﾗｰ or T3 ﾀｲﾑｱｳﾄなど）

【説明】

S1F17 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S1F18 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が onlack に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。（T3 ﾀｲﾑｱｳﾄも含む）

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

17. 3 class_SendS2F31 クラス - Date and Time Set Requst

S2F31 メッセージを送信し、S2F32 応答メッセージを受信するためのクラスです。
S2F31 は日付時刻設定要求のメッセージです。

17. 3. 1 コンストラクタ

下表の通りです。

	名前	説明
1	public class_SendS2F31()	ｲﾝｽﾀﾝｽを生成します。

17. 3. 2 プロパティ

なし。

17. 3. 3 メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public int SendS2F31()	S2F31 ﾒｯｾｰｼﾞを送信します。
2	public int SendS2F31_wait()	S2F31 ﾒｯｾｰｼﾞを送信し、S2F32 を受信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。

17．3．3．1　SendS2F31()

S2F31 メッセージの送信要求をします。
応答メッセージ S2F32 の情報は callback 関数の引数に渡されます。

【構文】

public int SendS2F31(string date_time, class_CALLBACK.callback_S2F32 callback, UInt32 upara)

【引数】

date_time

日付時刻を文字列で以下のように表現します。

“YYYYMMDDhhmmsscc”

年　月日時分秒 10ms 単位

2019 年 4 月 1 日 13:23:25.50 は “2019040113232550” のように表現します。

callback

S2F31 送信後、2 次メッセージを受信した時に呼び出される callback 関数です。

upara

ユーザパラメータ情報です。コールバックされる際に callback 関数の引数として渡されます。
要求とコールバック間のタグ情報として使用できます。

【戻り値】

返却値	意　味
0	要求が受け入れられた。
(-1)	エンコードに失敗した。または、要求が受け入れられなかった。

【説明】

引数 date_time（日付時刻データ）から S2F31 メッセージを生成し、それを相手装置へ送信するように要求します。
要求がエンジンによって受け入れられたときは 0 が、受け入れられなかった場合は(-1)が返却されます。
送信後、受信した応答メッセージの tiack を引数にして callback 関数を呼び出します。
callback 関数からは 0 を返却してください。

【callback の構文】

```
public delegate int callback_S2F32(
    int end_status,                 // 終了状態ｺｰﾄﾞ
    int tiack,                      // S2F32 に含まれる tiack です。
    uint upara                      // ﾕｰｻﾞ ﾊﾟﾗﾒｰﾀ(送信要求ﾒｿｯﾄﾞで指定された upara)
)
```

end_status に返却される値は、=0 の場合は正常終了を意味します。

end_status	意　味
0	送受信が正常に完了した。
-14	T3 ﾀｲﾑｱｳﾄを検出し、応答ﾒｯｾｰｼﾞを受信できなかった。
(-1)	送信できなかった。

17．３．３．２　SendS2F31_wait()

S2F31 メッセージの送信要求をし、引き続き応答メッセージも受信します。

【構文】

public int Send_wait(string date_time, ref int tiack)

【引数】

date_time

日付時刻を文字列で以下のように表現します。

“YYYYMMDDhhmmsscc”

年　月日時分秒 10ms 単位

2019 年 4 月 1 日 13:23:25.50 は “2019040113232550” のように表現します。

tiack

S2F32 応答メッセージの ACK 情報保存用です。

【戻り値】

返却値	意　味
0	正常に送受信した。
＜ 0	エンコードに失敗した。または、送受信ｴﾗｰが発生した。(送信ｴﾗｰ or T3ﾀｲﾑｱｳﾄなど)

【説明】

引数 date_time（日付時刻データ）から S2F31 メッセージを生成し、それを相手装置へ送信するように要求します。

送信要求の後、S2F32 応答メッセージを待機します。受信が終了したら、上の【戻り値】に示した値が返却されます。ei=0 の場合は、受信した応答 ACK が tiack に保存され渡されます。

応答メッセージを受信するまでプログラムはブロック（待ち）状態になります。

送信または、受信エラーを検出した場合は、ei ＜ 0 の値を返却します。(T3 ﾀｲﾑｱｳﾄも含む)

T3 タイムアウトが発生した場合、DSHDR2 HSMS ドライバーに設定されている T3 プロトコルタイムアウト時間だけブロックされることになります。

18. ユーザ固有 SECS-II メッセージの送信ならびに応答メッセージの送信

ユーザ固有メッセージの送信のために次の３つのメソッドが EngAPI クラスに準備されています。

- SendReuest() メソッド　　(非ブロックモードでの送受信)
- SendReuest_wait()メソッド(ブロックモードでの送受信)
- SendResponse()メソッド

これらは、static なメソッドになっていますので、EngAPI.SendReuest(..)のように呼び出してください。

18. １　SendReqSxFy クラス – ユーザ仕様メッセージの送信

18. １. １　コンストラクター

なし。　(すべてのメソッドが static です。)

18. １. ２　プロパティ W

なし

18. １. ３　メソッド

本クラスのメソッドは次の通りです。

	名前	説明
1	public static int SendRequest()	１次メッセージを非ブロックモードで送信します。
2	public static int SendRequest_wait()	１次メッセージをブロックモードで送信します。 ﾌﾟﾛｸﾞﾗﾑは応答受信までﾌﾞﾛｯｸされます。S10F1

18. 1. 3. 1 SendReuest()

GEM でサポートされていない、あるいは、当エンジンで標準サポートされていないユーザ独自の 1 次メッセージの送信を行います。本メソッドは非ブロックモードのメッセージの送信になります。

構文】

```
public static int SendRequest( ref DSHMSG smsg,
                        class_CALLBACK.callback_send_request callback, UInt32 upara)
```

【引数】

smsg

SECS-Ⅱメッセージ情報が格納されている構造体のポインタです。
メッセージの組立ては、ユーザが行います。

callback

SendReuest()が終了したときに呼び出されるコールバック関数(イベントハンドラー)です。

upara

ユーザが callback で指定した関数が呼び出された際に、引数で渡して欲しいデータです。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
< 0	SendReuest 要求に失敗した。

【説明】

本メソッドはユーザが組み立てた smsg に保存されている 1 次メッセージの送信を行います。そして、受信した応答メッセージは callback の引数 DSHMSG rmsg でユーザに報せます。

本メソッドは、static の関数として準備されていますので、次のコーディングで使用できます。

```
EngAPI.SendReuest(.... )
```

SendReuest()メソッドがエンジンに受け付けられた場合返却値 0 が、受け入れられなかった場合は(-1)が返却されます。

応答メッセージを受信できたら、callback によって指定された関数を呼び出します。その際、送信結果と受信メッセージが引数として与えられます。

ユーザは、本メソッドを実行する前に、smsg 構造体内に 1 次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストです。詳しくは、プログラミング例を参考にしてください。

【終了通知関数】

SendReuest()メソッドに対する callback の書式は、**class_CALLBACK クラス**に次のように定義されています。

```
public delegate int callback_SendReuest(int end_status, ref DSHMSG rmsg, uint upara);
```

end_status ： SendReuest()の処理結果です。 0 であれば正常に終了です。
(-1)であればエラー終了です。

rmsg　　：応答メッセージ情報が格納されている構造体のポインタです。
　　SendReuest()メソッドの引数で与えられた構造体のポインタです。
upara　　：SendReuest()メソッドで与えられた upara の値が渡されます。

【例】 S7F5 を送信し、S7F6 を受信する処理

```
private void SendS7F5( string ppid )
    int ei = 0;
    DSHMSG smsg = new DSHMSG();                          // 送信用構造体
    IntPtr buff = Marshal.AllocCoTaskMem(1024);          // ﾒｯｾｰｼﾞ Text 用ﾊﾞｯﾌｧ
    smsg.wbit = 1;                                       // Wait bit=1
    smsg.stream = 7;                                     // S7
    smsg.function = 5;                                   // F5
    while (true)
    {
        smsg.buffer = buff;                              // buff ptr 設定
        smsg.length = 1024;                              // buff size 設定

        HSMS.D_InitItemPut(ref smsg);                    // smsg 構造体の put のための初期化

        ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_L, IntPtr.Zero, 1);     // L-1 ｾｯﾄ
        if (ei < 0) break;

        ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_A, .ppid, ppid.Length); // PPID  ｾｯﾄ
        break;
    }
    if (ei < 0)
    {
        DshLog.log("  !! Message setup error¥n");      // 組立てｴﾗｰ
        return;
    }
    ei = EngAPI.Send_request(ref smsg, cback_request_S7F5, 705); // 送信
    if (ei < 0)
    {
        DshLog.log("  !! Send_request() error¥n");
    }
    Marshal.FreeCoTaskMem(buff);                         // buff ﾒﾓﾘ開放
}
```

送受信完了で呼び出される callback 関数

```
private static int callback_Send_request_S7F5(int end_status, ref DSHMSG rmsg, uint upara)
{
    string ppid ="";
    string ppbody ="";
    IntPtr ptr = Marshal.AllocCoTaskMem( 1024 );                    // dataitem 値取得用ﾊﾞｯﾌｧ

    DshLog.log("  ! SendRequest callback() end_status = " + end_status.ToString() + "¥r¥n");
    if (end_status == 0)
    {
        formid.fm.OutLog(" APP S" + rmsg.stream.ToString() + "F" + rmsg.function.ToString() + " rcvd");
        formid.fm.OutLog("     length=" + rmsg.length.ToString());
        HSMS.D_InitItemGet(ref rmsg);                                          // rmsg 初期化
        int n = 0;
        int ei = 0;
        while( true )
        {
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_L, IntPtr.Zero, 0);   / L-2
            if ( n != 2 ){
                ei = (-1); break;
            }
            n = HSMS.D_GetItem( ref rmsg, HSMS.ICODE_A, ptr, 1024 );      // PPID
            if ( n < 0 ){
                ei = (-1); break;
            }
            ppid = DshLib.DshPtrToString(ptr, 1024, n);
            n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_A, ptr, 1024);        // PPBODY
            if (n < 0){
                ei = (-1); break;
            }
            ppbody = DshLib.DshPtrToString(ptr, 1024, n);
            break;
        }
        if (ei == 0){
            DshLog.log("    ppid     = " + ppid + "¥n");
            DshLog.log("    ppbody   = " + ppbody + "¥n");
        }
        else{
            DshLog.log("  !! Message format error" + "¥n");
        }
        break;
    }
    Marshal.FreeCoTaskMem(ptr);                                            // ptr ﾒﾓﾘ開放
    return 0;
}
//----- callback 用 instance
private static DshCallback.callback_Send_request cback_request_S7F5 =
                                new DshCallback.callback_Send_request(callback_Send_request_S7F5);
```

18．1．3．2　SendReuest_wait()

GEM でサポートされていない、あるいは、エンジンで標準サポートされていないユーザ独自の 1 次メッセージをブロックモードで送信を行います。

構文】

```
public static int SendReuest_wait(ref DSHMSG smsg, ref DSHMSG rmsg)
```

【引数】

smsg

SECS-Ⅱメッセージ情報が格納されている構造体のポインタです。
メッセージの組立ては、ユーザが行います。

rmsg

応答 2 次メッセージ情報を格納するための構造体のポインタです。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
＜ 0	SendReuest_wait 送受信に失敗した。

【説明】

本メソッドはユーザが組み立てた smsg に格納されている 1 次メッセージの送信を行います。そして、受信した応答メッセージを rmsg に格納します。

本メソッドは、SendReuest()との違いは、SendReuest()は非ブロックモードの送受信であり、SendReuest_wait()は、ブロックモードでの送受信になります。
ブロックモードでは、応答メッセージの受信までプログラムはブロック（待ち）状態になります。

本メソッドは、static の関数として準備されていますので、インスタンスを生成しないで次のコーディングで使用できます。

```
EngAPI.SendReuest_wait(.... )
```

SendReuest_wait()メソッドが正常に完了したときは返却値 0 で戻ります。異常を検出した場合は負の値（＜ 0 ）が返却されます。

ユーザは、本メソッドを実行する前に、smsg 構造体内に 1 次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストなどです。詳しくは、次ページのプログラミング例を参考にしてください。

それから、受信メッセージの処理が終了したら、rmsg の中にメッセージ格納用に使用されたメモリの開放を必ず実行してください。実行しないとメモリリークが発生します。
HSMS_LIB.free_DSHMSG(ref rmsg) メソッドを使って開放します。

```
HSMS_LIB.free_DSHMSG ( ref rmsg );   // rmsg 内のﾊﾞｯﾌｧﾒﾓﾘを開放する。
```

【例】　S7F5 を送信し、S7F6 を受信する処理

```
private void SendS7F5_wait( string ppid )
{
    int ei = 0;
    DSHMSG smsg = new DSHMSG();                          // 送信用構造体
    DSHMSG rmsg = new DSHMSG();                          // 受信用構造体
    IntPtr buff = Marshal.AllocCoTaskMem(1024);          // ﾒｯｾｰｼﾞ Text 用ﾊﾞｯﾌｧ
    smsg.wbit = 1;                                       // Wait bit=1
    smsg.stream = 7;                                     // S7
    smsg.function = 5;                                   // F5
    smsg.buffer = buff;                                  // buff ptr 設定
    smsg.length = 1024;                                  // buff size 設定
    HSMS.D_InitItemPut(ref smsg);                        // smsg 構造体の put のための初期化
    ei = HSMS.D_PutItem(ref smsg, HSMS.ICODE_L, IntPtr.Zero, 1);      // L-1 ｾｯﾄ
    ei |= HSMS.D_PutItem(ref smsg, HSMS.ICODE_A, .ppid, ppid.Length); // PPID  ｾｯﾄ
    if (ei < 0){
        DshLog.log("  !! Message setup error¥n");                  // 組立てｴﾗｰ
        Marshal.FreeCoTaskMem(buff);
        return;
    }
    ei = EngAPI.Send_request_wait(ref smsg, ref rmsg);           // 送受信
    if (ei < 0){
        DshLog.log("  !! Send_request_wait() error¥n");{
    }else{
       IntPtr ptr = Marshal.AllocCoTaskMem(1024);
       HSMS.D_InitItemGet(ref rmsg);                             // rmsg 初期化
       int n = 0; int ei = 0;
       while( true ){
           n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_L, IntPtr.Zero, 0);       // L-2
           if ( n != 2 ){
               ei = (-1);
               break;
           }
           n = HSMS.D_GetItem( ref rmsg, HSMS.ICODE_A, ptr, 1024 );  // PPID
           if ( n < 0 ){
               ei = (-1); break;
           }
           ppid = DshLib.DshPtrToString(ptr, 1024, n);
           n = HSMS.D_GetItem(ref rmsg, HSMS.ICODE_A, ptr, 1024);  // PPBODY
           if (n < 0){
               ei = (-1); break;
           }
           ppbody = DshLib.DshPtrToString(ptr, 1024, n);
           break;
       }
       if (ei == 0) {
           DshLog.log("    ppid     = " + ppid + "¥n");
           DshLog.log("    ppbody   = " + ppbody + "¥n");
       }else{
           DshLog.log("   !! Message format error" + "¥n");
       }
       HSMS_LIB.free_DSHMSG ( ref rmsg );  // !!  これを必ず実行すること
       Marshal.FreeCoTaskMem(ptr);                                   // ptr ﾒﾓﾘ開放
    }
    Marshal.FreeCoTaskMem(buff);                                     // buff ﾒﾓﾘ開放
}
```

18. ２　SendResponse() – 応答メッセージの送信

２次メッセージの応答送信を行います。

18.

構文】

```
public static int SendResponse(ref DSHMSG rmsg, uint trid)
```

【引数】

rmsg

SECS-Ⅱ応答メッセージ情報が格納されている構造体のポインタです。
メッセージの組立ては、ユーザが行います。

trid

１次メッセージ受信時に受けとったトランザクション ID です。

【戻り値】

返却値	意　味
0	正常に要求が受付された。
＜ 0	SendResponse 要求に失敗した。

【説明】

本メソッドはユーザが組み立てた rmsg に保存されている２次メッセージの送信を行います。
その際、引数の trid を付けて送信します。

本メソッドは、static の関数として準備されていますので、インスタンスを生成いないで次のコーディングで使用できます。

```
EngAPI.SendResponse(.... )
```

SendResponse()メソッドがエンジンに受け付けられた場合、返却値 0 で戻ります。受け入れられなかった場合は(-1)が返却されます。

ユーザは、本メソッドを実行する前に、rmsg 構造体内に２次メッセージを組立てセットしなければなりません。ストリーム、ファンクション、そしてテキストなどです。詳しくは、プログラミング例を参考にしてください。

本メソッドは、エンジンに応答メッセージの送信を要求し、それが受付されてから後、送信が終了しても特に通知はありません。

【例】　S10F3 を受信した後、S10F4 を SendResponse を使って送信します。

```
public static void S10F1(int eqid, uint trid, ref DSHMSG smsg)
{

   // <ここで、S10F3 の処理を行う。
   // 以下、S10F4 ﾒｯｾｰｼﾞを準備し、応答送信します。

    int ackc10 = 0;

    DSHMSG rmsg = new DSHMSG();                                  // 2ｼﾞﾒｯｾｰｼﾞ情報格納構造体
    rmsg.stream = 10;                                            // S10
    rmsg.function = 4;                                           // F4
    rmsg.wbit = 0;                                               // w-bit = 0
    IntPtr buff = Marshal.AllocCoTaskMem(128);                   // text 用ﾊﾞｯﾌｧﾒﾓﾘ確保
    rmsg.buffer = buff;
    rmsg.length = 128;
    HSMS.D_InitItemPut(ref rmsg);                                // rmsg 構造体初期化
    HSMS.D_PutItem(ref rmsg, HSMS.ICODE_B, ref ackc10, 1);       // ackc10 を設定

    EngAPI.Send_response(trid, ref rmsg);                        // 応答送信
```