

pytelicam ライブラリ (Python Library)

ユーザーガイド

Version 1.1.1 (2024/12/13)

東芝テリー株式会社

改善の為予告なく変更することがありますので、最新の取扱説明書にて機能をご確認ください

DAA01622E

目次

1. はじめに.....	9
2. 構成.....	9
3. システム要件.....	10
4. インストール方法.....	10
5. アンインストール方法.....	11
6. ライブラリの使用方法.....	11
6.1. クラス階層.....	11
6.2. 一般的な使用方法.....	12
6.3. エラー処理.....	13
7. クラスの説明.....	14
7.1. 基本関数.....	15
7.1.1. get_camera_system 関数.....	15
7.2. CameraSystem クラス (基幹クラス).....	16
7.2.1. terminate 関数.....	17
7.2.2. get_information 関数.....	17
7.2.3. get_num_of_cameras 関数.....	17
7.2.4. get_camera_information 関数.....	18
7.2.5. create_device_object 関数.....	18
7.2.6. create_device_object_from_info 関数.....	19
7.2.7. create_device_object_from_ip_address 関数.....	20
7.2.8. create_signal 関数.....	21
7.2.9. close_signal 関数.....	21
7.2.10. wait_for_signal 関数.....	22
7.2.11. set_signal 関数.....	23
7.2.12. reset_signal 関数.....	23
7.2.13. register_cti_file 関数.....	24
7.3. CameraDevice クラス (カメラ制御).....	25
7.3.1. get_information 関数.....	27
7.3.2. open 関数.....	27
7.3.3. close 関数.....	28
7.3.4. read_register 関数.....	28
7.3.5. write_register 関数.....	29
7.3.6. reset_port 関数.....	30
7.3.7. get_heartbeat 関数.....	30
7.3.8. set_heartbeat 関数.....	31
7.3.9. save_parameter 関数.....	32
7.3.10. load_parameter 関数.....	32
7.3.11. get_last_genicam_error 関数.....	33
7.4. CameraStream クラス (ストリーム制御).....	34
7.4.1. open 関数.....	36
7.4.2. close 関数.....	38
7.4.3. start 関数.....	38
7.4.4. stop 関数.....	39
7.4.5. abort 関数.....	39
7.4.6. get_current_buffer_index 関数.....	39
7.4.7. get_next_image 関数.....	40
7.4.8. get_next_image_with_trigger 関数.....	41
7.4.9. get_current_buffered_image 関数.....	42
7.4.10. get_buffered_image 関数.....	43
7.4.11. chunk_attach_buffer 関数.....	44
7.4.12. chunk_update_buffer 関数.....	45
7.4.13. chunk_check_buffer_layout 関数.....	45
7.4.14. set_callback_image_acquired 関数.....	46

7.4.15. reset_callback_image_acquired 関数	47
7.4.16. set_callback_image_error 関数	47
7.4.17. reset_callback_image_error 関数	48
7.4.18. set_callback_buffer_busy 関数	49
7.4.19. reset_callback_buffer_busy 関数	50
7.5. CameraEvent クラス (イベント制御)	51
7.5.1. open 関数	52
7.5.2. close 関数	54
7.5.3. activate 関数	54
7.5.4. deactivate 関数	55
7.5.5. get_event_data 関数	56
7.5.6. clear_event_data 関数	57
7.5.7. get_queueing_count 関数	57
7.5.8. get_lost_count 関数	57
7.6. CameraControl クラス	58
7.6.1. ImageFormatControl	65
7.6.1.1. get_image_format_selector	65
7.6.1.2. set_image_format_selector	66
7.6.2. Scalable	67
7.6.2.1. get_sensor_width	67
7.6.2.2. get_sensor_height	68
7.6.2.3. get_roi	68
7.6.2.4. set_roi	69
7.6.2.5. get_width_min_max	69
7.6.2.6. get_width	70
7.6.2.7. set_width	70
7.6.2.8. get_height_min_max	71
7.6.2.9. get_height	71
7.6.2.10. set_height	72
7.6.2.11. get_offset_x_min_max	73
7.6.2.12. get_offset_x	73
7.6.2.13. set_offset_x	74
7.6.2.14. get_offset_y_min_max	74
7.6.2.15. get_offset_y	75
7.6.2.16. set_offset_y	75
7.6.3. Binning	76
7.6.3.1. get_binning_horaizontal_min_max	76
7.6.3.2. get_binning_horizontal	77
7.6.3.3. set_binning_horizontal	78
7.6.3.4. get_binning_vertical_min_max	79
7.6.3.5. get_binning_vertical	79
7.6.3.6. set_binning_vertical	80
7.6.4. Decimation	81
7.6.4.1. get_decimation_horizontal_min_max	81
7.6.4.2. get_decimation_horizontal	82
7.6.4.3. set_decimation_horizontal	82
7.6.4.4. get_decimation_vertical_min_max	83
7.6.4.5. get_decimation_vertical	83
7.6.4.6. set_decimation_vertical	84
7.6.5. Reverse	85
7.6.5.1. get_reverse_x	85
7.6.5.2. set_reverse_x	86
7.6.5.3. get_reverse_y	86
7.6.5.4. set_reverse_y	87
7.6.6. PixelFormat	88
7.6.6.1. get_pixel_format	88
7.6.6.2. set_pixel_format	89
7.6.7. TestPattern	90
7.6.7.1. get_test_pattern	90
7.6.7.2. set_test_pattern	91

7.6.8. AcquisitionControl.....	92
7.6.8.1. get_stream_payload_size	92
7.6.8.2. get_stream_enable	92
7.6.8.3. get_acquisition_frame_count_min_max	93
7.6.8.4. get_acquisition_frame_count	93
7.6.8.5. set_acquisition_frame_count	94
7.6.8.6. get_acquisition_frame_rate_control	95
7.6.8.7. set_acquisition_frame_rate_control	96
7.6.8.8. get_acquisition_frame_rate_min_max	97
7.6.8.9. get_acquisition_frame_rate	97
7.6.8.10. set_acquisition_frame_rate	98
7.6.8.11. execute_acquisition_start	99
7.6.8.12. get_high_framerate_mode	100
7.6.8.13. set_high_framerate_mode	101
7.6.9. ImageBuffer	102
7.6.9.1. get_image_buffer_mode	102
7.6.9.2. set_image_buffer_mode	103
7.6.9.3. get_image_buffer_frame_count	103
7.6.9.4. execute_image_buffer_read	104
7.6.10. TriggerControl	105
7.6.10.1. get_trigger_mode	105
7.6.10.2. set_trigger_mode	106
7.6.10.3. get_trigger_sequence	107
7.6.10.4. set_trigger_sequence	108
7.6.10.5. get_trigger_source	108
7.6.10.6. set_trigger_source	109
7.6.10.7. get_trigger_additional_parameter_min_max	110
7.6.10.8. get_trigger_additional_parameter	111
7.6.10.9. set_trigger_additional_parameter	112
7.6.10.10. get_trigger_delay_min_max	113
7.6.10.11. get_trigger_delay	113
7.6.10.12. set_trigger_delay	114
7.6.10.13. execute_software_trigger	114
7.6.10.14. get_trigger_activation	115
7.6.10.15. set_trigger_activation	116
7.6.11. ExposureTime	117
7.6.11.1. get_exposure_time_control	117
7.6.11.2. set_exposure_time_control	118
7.6.11.3. get_exposure_time_min_max	119
7.6.11.4. get_exposure_time	119
7.6.11.5. set_exposre_time	120
7.6.11.6. get_short_exposure_mode	121
7.6.11.7. set_short_exposure_mode	122
7.6.12. DigitalloControl	123
7.6.12.1. get_line_mode_all	123
7.6.12.2. set_line_mode_all	124
7.6.12.3. get_line_mode	125
7.6.12.4. set_line_mode	126
7.6.12.5. get_line_inverter_all	127
7.6.12.6. set_line_inverter_all	128
7.6.12.7. get_line_inverter	129
7.6.12.8. set_line_inverter	130
7.6.12.9. get_line_status_all	131
7.6.12.10. get_line_status	131
7.6.12.11. get_user_output_value_all	132
7.6.12.12. set_user_output_value_all	133
7.6.12.13. get_user_output_value	134
7.6.12.14. set_user_output_value	135
7.6.12.15. get_line_source	136
7.6.12.16. set_line_source	137
7.6.13. AntiClitch / AntiChattering	138
7.6.13.1. get_anti_glitch_min_max	139

7.6.13.2. get_anti_glitch	139
7.6.13.3. set_anti_glitch	140
7.6.13.4. get_anti_chattering_min_max	141
7.6.13.5. get_anti_chattering	141
7.6.13.6. set_anti_chattering	142
7.6.14. TimerControl	143
7.6.14.1. get_timer_duration_min_max	143
7.6.14.2. get_timer_duration	144
7.6.14.3. set_timer_duration	144
7.6.14.4. get_timer_delay_min_max	145
7.6.14.5. get_timer_delay	145
7.6.14.6. set_timer_delay	146
7.6.14.7. get_timer_trigger_source	146
7.6.14.8. set_timer_trigger_source	147
7.6.15. Gain	148
7.6.15.1. get_gain_min_max	148
7.6.15.2. get_gain	149
7.6.15.3. set_gain	149
7.6.15.4. get_gain_auto	150
7.6.15.5. set_gain_auto	150
7.6.16. BlackLevel	151
7.6.16.1. get_black_level_min_max	151
7.6.16.2. get_black_level	152
7.6.16.3. set_black_level	152
7.6.17. Gamma	153
7.6.17.1. get_gamma_min_max	153
7.6.17.2. get_gamma	154
7.6.17.3. set_gamma	154
7.6.18. WhiteBalance	155
7.6.18.1. get_balance_ratio_min_max	155
7.6.18.2. get_balance_ratio	156
7.6.18.3. set_balance_ratio	157
7.6.18.4. get_balance_white_auto	157
7.6.18.5. set_balance_white_auto	158
7.6.19. Hue	159
7.6.19.1. get_hue_min_max	159
7.6.19.2. get_hue	160
7.6.19.3. set_hue	160
7.6.20. Saturation	161
7.6.20.1. get_saturation_min_max	161
7.6.20.2. get_saturation	162
7.6.20.3. set_saturation	163
7.6.21. Sharpness	164
7.6.21.1. get_sharpness_min_max	164
7.6.21.2. get_sharpness	165
7.6.21.3. set_sharpness	165
7.6.22. ColorCorrectionMatrix	166
7.6.22.1. get_color_correction_matrix_min_max	167
7.6.22.2. get_color_correction_matrix	168
7.6.22.3. set_color_correction_matrix	169
7.6.23. LUT Control	170
7.6.23.1. get_lut_enable	170
7.6.23.2. set_lut_enable	171
7.6.23.3. get_lut_value	172
7.6.23.4. set_lut_value	173
7.6.24. UserSetControl	174
7.6.24.1. execute_user_set_load	174
7.6.24.2. execute_user_set_save	175
7.6.24.3. execute_user_set_quick_save	176
7.6.24.4. get_user_set_default	177
7.6.24.5. set_user_set_default	178
7.6.24.6. execute_user_set_save_and_set_default	179

7.6.25. SequentialShutterControl.....	180
7.6.25.1. get_sequential_shutter_enable.....	180
7.6.25.2. set_sequential_shutter_enable.....	181
7.6.25.3. get_sequential_shutter_terminate_at_min_max.....	181
7.6.25.4. get_sequential_shutter_terminate_at.....	182
7.6.25.5. set_sequential_shutter_terminate_at.....	182
7.6.25.6. get_sequential_shutter_index_min_max.....	183
7.6.25.7. get_sequential_shutter_entry_min_max.....	183
7.6.25.8. get_sequential_shutter_entry.....	184
7.6.25.9. set_sequential_shutter_entry.....	185
7.6.26. UserDefinedName.....	186
7.6.26.1. get_user_defined_name.....	186
7.6.26.2. set_user_defined_name.....	187
7.6.27. Chunk.....	188
7.6.27.1. get_chunk_mode_active.....	188
7.6.27.2. set_chunk_mode_active.....	189
7.6.27.3. get_chunk_enable.....	190
7.6.27.4. set_chunk_enable.....	191
7.6.27.5. get_chunk_user_area_length.....	191
7.6.27.6. get_chunk_user_area_table.....	192
7.6.27.7. set_chunk_user_area_table.....	192
7.6.28. FrameSynchronization.....	193
7.6.28.1. get_frame_synchronization.....	193
7.6.28.2. set_frame_synchronization.....	194
7.7. GenApiWrapper クラス (カメラ制御用サブセット).....	195
7.7.1. INode 型関数.....	197
7.7.1.1. get_node_type 関数.....	197
7.7.1.2. get_access_mode 関数.....	198
7.7.1.3. get_visibility 関数.....	199
7.7.1.4. get_caching_mode 関数.....	200
7.7.1.5. get_description 関数.....	201
7.7.1.6. get_tool_tip 関数.....	202
7.7.1.7. get_representation 関数.....	203
7.7.1.8. get_unit 関数.....	204
7.7.2. ICategory 型関数.....	205
7.7.2.1. get_num_of_features 関数.....	205
7.7.2.2. get_feature_name 関数.....	206
7.7.2.3. get_available_feature_names 関数.....	207
7.7.3. 共通関数.....	208
7.7.3.1. get_feature_value 関数.....	208
7.7.3.2. set_feature_value 関数.....	209
7.7.4. IInteger 型関数.....	210
7.7.4.1. get_int_min 関数.....	210
7.7.4.2. get_int_max 関数.....	211
7.7.4.3. get_int_inc 関数.....	212
7.7.4.4. get_int_value 関数.....	213
7.7.4.5. set_int_value 関数.....	214
7.7.5. IFloat 型関数.....	215
7.7.5.1. get_float_min 関数.....	215
7.7.5.2. get_float_max 関数.....	216
7.7.5.3. get_float_has_inc 関数.....	217
7.7.5.4. get_float_inc 関数.....	218
7.7.5.5. get_float_display_notation 関数.....	219
7.7.5.6. get_float_display_precision 関数.....	220
7.7.5.7. get_float_value 関数.....	221
7.7.5.8. set_float_value 関数.....	222
7.7.6. IBoolean 型関数.....	223
7.7.6.1. get_bool_value 関数.....	223

7.7.6.2. set_bool_value 関数	224
7.7.7. IEnumeration 型	225
7.7.7.1. get_enum_int_value 関数	225
7.7.7.2. set_enum_int_value 関数	226
7.7.7.3. get_enum_str_value 関数	227
7.7.7.4. set_enum_str_value 関数	228
7.7.7.5. get_available_enum_entry_names 関数	229
7.7.7.6. get_num_of_enum_entries 関数	230
7.7.7.7. get_enum_entry_access_mode 関数	231
7.7.7.8. get_enum_entry_int_value 関数	232
7.7.7.9. get_enum_entry_str_value 関数	233
7.7.8. ICommand 型関数	234
7.7.8.1. execute_command 関数	234
7.7.8.2. get_command_is_done 関数	235
7.7.9. IString 型関数	236
7.7.9.1. get_str_value 関数	236
7.7.9.2. set_str_value 関数	237
7.7.10. その他関数	238
7.7.10.1. get_access_module 関数	238
7.7.10.2. set_access_module 関数	238
7.8. SignalHandle クラス	239
7.9. CamSystemInfo クラス	239
7.10. CameraInfo クラス	240
7.11. ImageData クラス	241
7.11.1. release 関数	242
7.11.2. get_ndarray 関数	243
7.11.3. get_memoryview 関数	243
7.12. EventData クラス	244
7.12.1. release 関数	245
7.12.2. get_ndarray 関数	246
7.12.3. get_memoryview 関数	246
7.13. PytelicamError クラス	247
8. 列挙型の説明	248
8.1. CamApiStatus 列挙型（ステータスコード）	250
8.2. CameraType 列挙型	252
8.3. CameraAccessMode 列挙型	252
8.4. CameraPixelFormat 列挙型	253
8.5. CameraAcquisitionMode 列挙型	254
8.6. CameraEventType 列挙型	255
8.7. NodeType 列挙型	255
8.8. NodeAccessMode 列挙型	256
8.9. NodeVisibility 列挙型	256
8.10. NodeCachingMode 列挙型	256
8.11. NodeRepresentation 列挙型	257
8.12. NodeDisplayNotation 列挙型	257
8.13. AccessModuleType 列挙型	257
8.14. OutputImageType 列挙型	258
8.15. BayerConversionMode 列挙型	258
8.16. CameraImageFormat 列挙型	258
8.17. CameraTestPattern 列挙型	259
8.18. CameraAcqFrameRateCtrl 列挙型	259
8.19. CameraImageBufferMode 列挙型	259
8.20. CameraTriggerSequence 列挙型	260
8.21. CameraTriggerSource 列挙型	261
8.22. CameraTriggerActivation 列挙型	261

8.23. CameraExposureTimeCtrl 列挙型	262
8.24. CameraLineSelector 列挙型	262
8.25. CameraLineSource 列挙型	263
8.26. CameraLineMode 列挙型	263
8.27. CameraTimerTriggerSource 列挙型	263
8.28. CameraGainAuto 列挙型	264
8.29. CameraBalanceRatioSelector 列挙型	264
8.30. CameraBalanceWhiteAuto 列挙型	264
8.31. CameraSaturationSelector 列挙型	265
8.32. CameraColorCorrectionMatrixSelector 列挙型	265
8.33. CameraUserSetSelector 列挙型	266
8.34. CameraChunkSelector 列挙型	266
8.35. CameraFrameSynchronization 列挙型	267
9. サンプルソース	268
10. その他	269
10.1. 免責事項	269
10.2. ライセンス	269
10.3. 改定履歴	271
10.4. お問い合わせ	271

1. はじめに

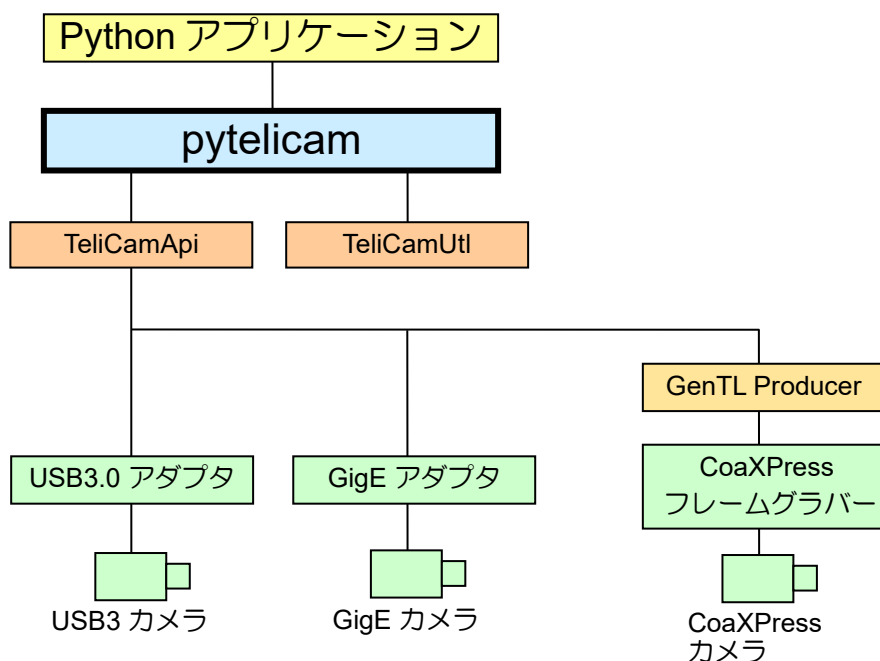
pytelicam ライブラリは、プログラミング言語 Python で TeliCamSDK を使用するための Python ライブラリです。

pytelicam ライブラリを使用することにより、TeliCamSDK で提供されているほぼすべての機能を Python 言語で使用することができます。

本ドキュメントの読者として、カメラを使用したシステムを構築するソフトウェア技術者を想定しています。

2. 構成

pytelicam ライブラリのソフトウェア構成は以下のとおりです。



構成	内 容
pytelicam	Python 言語用ライブラリ
TeliCamApi	ネイティブアプリケーション開発用関数ライブラリ
TeliCamUtl	画像ハンドリングユーティリティ関数ライブラリ
GenTL Producer	フレームグラバー メーカーにより提供される GenTL Producer ライブラリ

3. システム要件

pytelicam ライブラリを使用するためには以下のソフトウェア がインストールされている必要があります。

Windows 64bit (win_amd64)	TeliCamSDK v4.0.4.1 以上
Linux 64bit (linux_x86_64) Linux ARM64 (linux_aarch64)	TeliCamSDK for Linux v4.0.4.1 以上

TeliCamSDK のインストール方法は、TeliCamSDK の "TeliCamSDK Start-up Guide" をご参照ください。

4. インストール方法

pytelicam ライブラリは、Python wheel(.whl) の形式で配布されます。 東芝テリーのホームページから必要なライブラリをダウンロードし、インストールしてください。

wheel ファイル名は以下のように定義されています。

{distribution}-{version}-{python tag}-{abi tag}-{platform tag}.whl

{distribution} は、"pytelicam" です。

{version} は、pytelicam ライブラリ のバージョンを示します。

{python tag}、{abi tag} と Python バージョンの関係は以下のとおりです。

例： Python 3.9.x の場合

- cp39-cp39

{platform tag} は以下のプラットフォームを示します。

win_amd64 : Windows 64bit

linux_x86_64 : Linux 64bit

pytelicam ライブラリのインストールは、pip コマンドにより実行します。

例： Python 3.9.x の場合

```
pip install pytelicam-1.1.1-cp39-cp39-win_amd64.whl
```

5. アンインストール方法

pytelicam ライブラリのアンインストールは、pip コマンドにより実行します。

例：

```
pip uninstall pytelicam
```

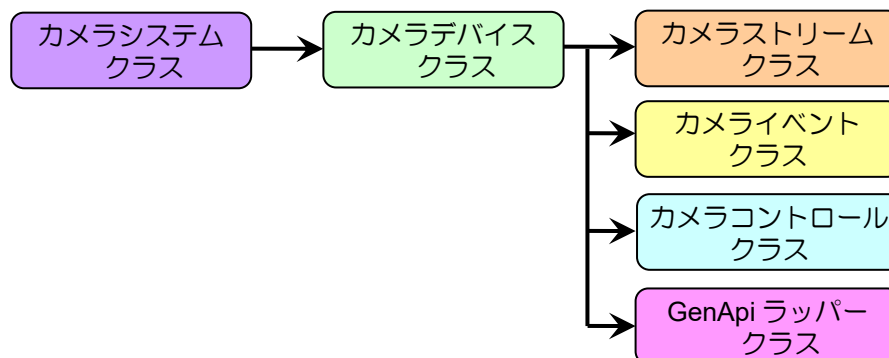
6. ライブラリの使用方法

pytelicam ライブラリは、TeliCamAPI のラッパーライブラリです。 構成は、TeliCamDNetAPI に類似しています。

基本的な使用方法や動作に関しては、TeliCamSDK の“TeliCamAPI Library Manual Jpn.pdf”を参照してください。

6.1. クラス階層

pytelicam ライブラリの主なクラス階層は以下のとおりです。



クラス	内 容
システムクラス	pytelicam ライブラリのルートとなるシステムクラス。
カメラデバイスクラス	カメラの全般的な管理を行うクラス。 このクラスはカメラストリーム、カメライベント、GenApi ラッパーの各クラスオブジェクトをメンバ変数として持ちます。
カメラストリームクラス	カメラから送られるストリーム（画像）を管理するクラス。
カメライベントクラス	カメラから送られるカメライベント通知（メッセージ）を管理するクラス。
カメラコントロールクラス	カメラの各機能個別制御用クラス。
GenApi ラッパークラス	カメラの各機能を制御するための GenICam GenApi モジュールをラップしたクラス。

6.2. 一般的な使用方法

1. pytelicam ライブラリをインポートします。

```
import pytelicam
```

2. pytelicam ライブラリを初期化し、[CameraSystem](#) クラスのオブジェクトを取得します。

```
cam_system = pytelicam.get_camera_system()
```

3. 接続されているカメラを列挙します。

```
cam_num = cam_system.get_num_of_cameras()
```

4. カメラの [CameraDevice](#) オブジェクトを取得します。

```
cam_device = cam_system.create_device_object()
```

5. カメラをオープンします。

```
cam_device.open()
```

6. ストリームインターフェースをオープンします。

```
cam_device.cam_stream.open()
```

7. 画像データの転送を開始します。

```
cam_device.cam_stream.start()
```

8. カメラからの画像データ([ImageData](#) オブジェクト)を取得します。 取得方法はいくつかありますが、ここでは次に到着する画像を取得します。

```
image_data = cam_device.cam_stream.get_next_image()
```

9. 取得した画像データを BGR24 フォーマットの NumPy 配列で取得します。

```
np_arr = image_data.get_ndarray(pytelicam.OutputImageType.Bgr24)
```

10. 画像の使用が終了したら、使用済の [ImageData](#) オブジェクトを開放します。開放せずに画像データの取り込みを繰り返した場合、本 API 内部のストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、画像を取得することができなくなります。

```
image_data.release()
```

11. 画像データの転送を停止します。

```
cam_device.cam_stream.stop()
```

12. ストリームインターフェースをクローズします。

```
cam_device.cam_stream.close()
```

13. カメラをクローズします。

```
cam_device.close()
```

14. [CameraSystem](#) クラスのオブジェクトの終了処理を実行し、アプリケーションを終了します。

```
cam_system.terminate()
```

6.3. エラー処理

pytelicam ライブラリは、戻り値のステータスコードでエラーを報告する関数と、例外でエラーを報告する関数があります。

戻り値にステータスコードを返す関数は、ステータスコードに `Success` が設定されていない場合でも pytelicam ライブラリにとっては正常に処理が行われている場合があります。

例えば、カメラが画像転送中に pytelicam ライブラリを使用してカメラの ROI を変更しようとしたとき、カメラは `NotWritable` エラーを返します。pytelicam ライブラリはステータスコードに [pytelicam.CamApiStatus.NotWritable](#) を設定しリターンしますが、pytelicam ライブラリ自身の処理としてはエラーなく正常に終了しています。

アプリケーションは、戻り値のステータスコードを確認し、分岐処理を行うことが期待されます。

例外でエラーを報告する関数は、カメラおよび API が正常に動作している場合は発生しないエラーを例外として送出します。

例外は [pytelicam.PytelicamError](#) が送出されます。

7. クラスの説明

pytelicam ライブラリで公開されるクラスは以下のとおりです。

[クラス]

	名 称	内 容
	CameraSystem	pytelicam ライブラリのルートとなるシステムクラス
	CameraDevice	カメラの全般的な管理を行うクラス
	CameraStream	カメラから送られるストリーム（画像）を管理するクラス
	CameraEvent	カメラから送られるカメライベント通知（メッセージ）を管理するクラス
	CameraControl	カメラの制御を行うクラス
	GenApiWrapper	カメラの各機能を制御するための GenICam GenApi モジュールをラップしたクラス
	SignalHandle	シグナルを管理するクラス
	CamSystemInfo	pytelicam ライブラリのシステム情報を提供するクラス
	CameraInfo	カメラ情報を提供するクラス
	ImageData	画像データを管理するクラス
	EventData	カメライベント通知（メッセージ）のデータを管理するクラス
	PytelicamError	pytelicam で例外が発生したときに送出されるクラス

[備考]

本マニュアルに記載されていないクラスおよび関数は、未サポートのクラスおよび関数です。

7.1. 基本関数

7.1.1. get_camera_system 関数

pytelicam ライブラリの初期化を実行し、[CameraSystem](#) オブジェクトを取得します。

[構文]

```
pytelicam.get_camera_system(camera_type)
```

[パラメータ]

パラメータ	内 容
camera_type (int)	使用するカメラのインターフェースタイプ。 pytelicam.CameraType を int 型で指定します。 引数を指定しない場合、東芝テリー製デバイスドライバを使用したすべてのタイプ (pytelicam.CameraType.All) が指定されます。 ※ GenTL インターフェースは含まれません

[戻り値]

作成された [CameraSystem](#) クラスのオブジェクト

[戻り値の型]

[pytelicam.CameraSystem](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数は CameraSystem オブジェクトを初期化し、pytelicam ライブラリを使用可能にする関数です。
pytelicam ライブラリの他の関数を使用する前に、本関数を 1 回だけ実行しておく必要があります。

東芝テリー製デバイスドライバを使用したすべてのインターフェースを使用する場合は、以下のようになります。(GenTL インターフェースは含まれません。)

```
cam_system = pytelicam.get_camera_system()
```

特定のインターフェースのみ使用する場合は、以下のようになります。

```
cam_system = pytelicam.get_camera_system(int(pytelicam.CameraType.U3v))
```

複数のインターフェースを使用する場合は、以下のようになります。

```
cam_system = pytelicam.get_camera_system( \
    int(pytelicam.CameraType.U3v) | \
    int(pytelicam.CameraType.Gev) | \
    int(pytelicam.CameraType.GenTL))
```

使用するインターフェースが決まっている場合は、引数を設定することで一部関数の実行速度が向上する可能性があります。

7.2. CameraSystem クラス (基幹クラス)

pytelicam ライブラリのルートとなるシステムクラスです。

[構文]

pytelicam.CameraSystem

[関数]

	名 称	内 容
⇒	terminate	CameraSystem オブジェクトの終了処理を実行します。
⇒	get_information	pytelicam ライブラリのシステム情報を取得します。
⇒	get_num_of_cameras	使用可能なカメラの数を取得します。
⇒	get_camera_information	検出したカメラの情報を取得します。
⇒	create_device_object	指定されたカメラインデックスの CameraDevice オブジェクトを作成します。
⇒	create_device_object_from_info	指定されたカメラ情報を持つカメラの CameraDevice オブジェクトを作成します。
⇒	create_device_object_from_ip_address	指定された IP アドレスが割り付けられたカメラの CameraDevice オブジェクトを作成します。
⇒	create_signal	シグナルを検知するためのシグナルオブジェクトを作成します。
⇒	close_signal	シグナルオブジェクトの使用を終了します。
⇒	wait_for_signal	指定されたシグナルオブジェクトの現在の状態を確認します。
⇒	set_signal	指定されたシグナルオブジェクトをシグナル状態にセットします。
⇒	reset_signal	指定されたシグナルオブジェクトを非シグナル状態にリセットします。
⇒	register_cti_file	使用する GenTL Producer を登録します。

7.2.1. terminate 関数

CameraSystem オブジェクトの終了処理を実行します。

[構文]

```
pytelicam.CameraSystem.terminate(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

CameraSystem オブジェクトを破棄する前に、一度だけ本関数を実行する必要があります。

7.2.2. get_information 関数

pytelicam ライブラリのシステム情報を取得します。

[構文]

```
pytelicam.CameraSystem.get_information(self)
```

[戻り値]

システム情報

[戻り値の型]

[pytelicam.CamSystemInfo](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.3. get_num_of_cameras 関数

使用可能なカメラのリストを本 API 内部に作成し、リスト内のカメラの数を返します。

[構文]

```
pytelicam.CameraSystem.get_num_of_cameras(self)
```

[戻り値]

検出したカメラの数

[戻り値の型]

int

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

カメラデバイスのオブジェクトを取得する前は、本関数が実行されている必要があります。
カメラの情報に変化があった場合は、再度本関数を実行する必要があります。

7.2.4. get_camera_information 関数

検出したカメラの情報を取得します。

[構文]

```
pytelicam.CameraSystem.get_camera_information(self, camera_index)
```

[パラメータ]

パラメータ	内 容
camera_index (int)	カメラのインデックス。 インデックスは、0 から get_num_of_cameras() で得られた "カメラ数 - 1" までの値が指定できます。

[戻り値]

カメラ情報

[戻り値の型]

[pytelicam.CameraInfo](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.5. create_device_object 関数

指定されたカメラインデックスのカメラの [CameraDevice](#) オブジェクトを作成します。

[構文]

```
pytelicam.CameraSystem.create_device_object(self, camera_index=0)
```

[パラメータ]

パラメータ	内 容
camera_index (int)	カメラのインデックス。 インデックスは、0 から get_num_of_cameras() で得られた "カメラ数 - 1" までの値が指定できます。

[戻り値]

作成された [CameraDevice](#) クラスのオブジェクト

[戻り値の型]

[pytelicam.CameraDevice](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.6. create_device_object_from_info 関数

パラメータで指定された情報を持つカメラの [CameraDevice](#) オブジェクトを作成します。

[構文]

```
pytelicam.CameraSystem.create_device_object_from_info(  
                                                    self,  
                                                    serial_no,  
                                                    model_name,  
                                                    user_defined_name)
```

[パラメータ]

パラメータ	内 容
serial_no (str)	カメラを特定するためのシリアル番号
model_name (str)	カメラを特定するためのモデル名
user_defined_name (str)	カメラを特定するためのユーザー定義情報

[戻り値]

作成された [CameraDevice](#) クラスのオブジェクト

[戻り値の型]

[pytelicam.CameraDevice](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

serial_no, model_name, user_defined_name の中でカメラの指定に使用しない引数は “ ” に設定してください。

```
cam_device = cam_system.create_device_object_from_info('0210001', 'BU406MC', '')
```

指定したパラメータを持つカメラが複数存在する場合、最初に見つかったカメラのオブジェクトを取得します。

7.2.7. create_device_object_from_ip_address 関数

パラメータで指定された IP アドレスが割り付けられた GigE カメラの [CameraDevice](#) オブジェクトを作成します。

[構文]

```
pytelicam.CameraSystem.create_device_object_from_ip_address (  
    self,  
    ip_string)
```

[パラメータ]

パラメータ	内 容
ip_string (str)	カメラを特定するための IPv4 アドレス。 IPv4 アドレスは文字列で指定します。 設定例: '192.168.0.16'

[戻り値]

作成された [CameraDevice](#) クラスのオブジェクト

[戻り値の型]

[pytelicam.CameraDevice](#)

[例外]

[pytelicam.PytelicamError](#): この API でエラーが発生したとき送出されます。

[備考]

本関数は、GigE カメラのみ有効です。

以下にコード例を示します。

```
cam_device = cam_system.create_device_object_from_ip_address('192.168.0.16')
```

7.2.8. create_signal 関数

シグナルを検知するためのシグナルオブジェクトを作成します。
シグナルオブジェクトは、カメラからの画像受信およびカメライベント（メッセージ）などのシグナルを検知するために使用します。

【構文】

`pytelicam.CameraSystem.create_signal(self)`

【戻り値】

シグナルオブジェクト

【戻り値の型】

[pytelicam.SignalHandle](#)

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.9. close_signal 関数

シグナルオブジェクトをクローズします。

【構文】

`pytelicam.CameraSystem.close_signal(self, signal_object)`

【パラメータ】

パラメータ	内 容
signal_object (pytelicam.SignalHandle)	シグナルオブジェクト

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.10. wait_for_signal 関数

指定されたシグナルオブジェクトの現在の状態を確認します。

シグナルオブジェクトがシグナル状態にセットされている場合は、直ちに制御を返します。このとき、シグナルオブジェクトは、非シグナル状態にリセットされます。

シグナルオブジェクトが非シグナル状態の場合、呼び出したスレッドはシグナルオブジェクトがシグナル状態にセットされるタイムアウト時間が経過するまで待機状態になります。

【構文】

```
pytelicam.CameraSystem.wait_for_signal(self, signal_object, milliseconds=5000)
```

【パラメータ】

パラメータ	内 容
signal_object (pytelicam.SignalHandle)	シグナルオブジェクト
milliseconds (int)	シグナル状態にセットされるまで待機するタイムアウト時間。(単位：ミリ秒) 0 以外を指定すると、この関数は指定されたシグナルオブジェクトがシグナル状態にセットされるタイムアウト時間が経過するまで待機します。 0 を指定すると、この関数は指定されたシグナルオブジェクトの状態にかかわらず、即座に制御を返します。 -1 または 0xFFFFFFFF を指定すると、指定されたシグナルオブジェクトがシグナル状態にセットされるまで待機します。

【戻り値】

実行結果を表すステータスコード。 戻り値は以下のとおりです。

CamApiStatus.Success : 指定されたシグナルオブジェクトがシグナル状態にセットされたことを意味します。

CamApiStatus.Timeout : タイムアウト時間が経過し、指定されたシグナルオブジェクトがシグナル状態にセットされなかったことを意味します。

CamApiStatus.Unsuccessful : 定義されていないエラーが発生したことを意味します。

【戻り値の型】

[pytelicam.CamApiStatus](#)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

7.2.11. set_signal 関数

指定されたシグナルオブジェクトをシグナル状態にセットします。

【構文】

```
pytelicam.CameraSystem.set_signal(self, signal_object)
```

【パラメータ】

パラメータ	内 容
signal_object (pytelicam.SignalHandle)	シグナルオブジェクト

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.12. reset_signal 関数

指定されたシグナルオブジェクトを非シグナル状態にリセットします。

【構文】

```
pytelicam.CameraSystem.reset_signal(self, signal_object)
```

【パラメータ】

パラメータ	内 容
signal_object (pytelicam.SignalHandle)	シグナルオブジェクト

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.2.13. register_cti_file 関数

使用する GenTL Producer (cti ファイル) を登録します。

pytelicam.get_camera_system() 関数で GenTL インターフェースを指定したときのみ有効です。

登録された GenTL Producer 以外はロードされないため、カメラ列挙処理を高速化することができます。

また、不具合が発生する特定の GenTL Producer を除外することができます。

[構文]

```
pytelicam.CameraSystem.register_cti_file(self, file_name)
```

[パラメータ]

パラメータ	内 容
file_name (str)	GenTL Producer ファイルの絶対パス (文字列)

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数は、pytelicam.get_camera_system() 関数を実行した直後に実行してください。 カメラオープン後に本関数を実行した場合は正常に処理が行われなくなる場合があります。

本関数を実行した場合、環境変数 GENICAM_GENTL64_PATH (32bit 版は GENICAM_GENTL32_PATH) で指定されたディレクトリに存在する GenTL Producer (cti ファイル) はロードされなくなります。

複数の GenTL Producer を登録することが可能です。

7.3. CameraDevice クラス (カメラ制御)

カメラの全般的な管理を行うクラスです。

[CameraSystem](#) クラスの [create_device_object\(\)](#) または [create_device_object_from_info\(\)](#) 関数で、本クラスのオブジェクトが取得できます。

【構文】

pytelicam.CameraDevice




【関数】

	名 称	内 容
⇒	get_information	カメラの情報を取得します。
⇒	open	カメラをオープンします。
⇒	close	カメラをクローズします。
⇒	read_register	カメラのレジスタからデータを取得します。
⇒	write_register	カメラのレジスタへデータを設定します。
⇒	reset_port	ホストアダプタ/ホストコントローラのポートリセットを実行します。
⇒	get_heartbeat	カメラのハートビート設定を取得します。
⇒	set_heartbeat	カメラのハートビート設定を変更します。
⇒	save_parameter	カメラのパラメータをセーブします。
⇒	load_parameter	カメラのパラメータをロードします。
⇒	get_last_genicam_error	GenICamError が発生したときの、エラー情報を取得します。

【メンバ】

	名 称	内 容
◆	cam_stream (pytelicam.CameraStream)	CameraStream オブジェクト。 カメラのストリームの設定を行い、画像データを取得するためのオブジェクトです。
◆	cam_event (pytelicam.CameraEvent)	CameraEvent オブジェクト。 カメラのイベント通知の設定を行い、カメライベント通知（メッセージ）を取得するためのオブジェクトです。
◆	cam_control (pytelicam.CameraControl)	CameraControl オブジェクト カメラ制御関数を使用し、カメラを制御するためのオブジェクトです。
◆	genapi (pytelicam.GenApiWrapper)	GenApiWrapper オブジェクト。 GenICam GenApi を使用し、カメラを制御するためのオブジェクトです。

[プロパティ]

	名 称	内 容
	is_open (bool)	カメラのオープン／クローズ状態を示す値。 True のときオープン状態、False のときクローズ状態です。
	is_support_iidc2 (bool)	カメラが IIDC2 規格に準拠しているかどうかを示す値。 True のとき準拠、False のとき非準拠です。
	camera_type (pytelicam.CameraType)	カメラのインターフェースタイプ。

7.3.1. get_information 関数

カメラの情報を取得します。

[構文]

```
pytelicam.CameraDevice.get_camera_information(self)
```

[戻り値]

カメラ情報

[戻り値の型]

[pytelicam.CameraInfo](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.3.2. open 関数

カメラをオープンします。

Windows 版では、他のアプリケーションが使用しているカメラもオープンすることができます。但し、複数のアプリケーションが同じストリームインターフェースを使用することはできません。

[構文]

```
pytelicam.CameraDevice.open(  
    self,  
    removed_signal=None,  
    access_mode=pytelicam.CameraAccessMode.Control)
```

[パラメータ]

パラメータ	内 容
removed_signal (pytelicam.SignalHandle)	カメラの切断を通知するためのシグナルオブジェクト。 カメラの切断を検知すると、シグナル状態に変化します。 カメラの切断を知る必要がない時は None を指定します。
access_mode (pytelicam.CameraAccessMode)	カメラのアクセスモード（特権）。 本パラメータは、GigE カメラ使用時以外は無視されます。 特別な理由がない場合は、パラメータを省略してください。

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

removed_signal は以下の場合にシグナル状態に設定されます。

- カメラケーブルの取り外しにより切断が検知されたとき
- 通信エラーにより切断が検知されたとき
- カメラの DeviceReset が実行されたとき
- [reset_port\(\)](#) 関数が実行されたとき

GigE カメラの場合、一定時間内にカメラとの通信タイムアウトが複数回連続発生したときにハートビートタイムエラーとなります。pytelicam は、ハートビートタイムアウトエラーが発生したときに通信が切断されたと判断します。

ハートビート設定値が False（無効）の場合、切断は検知されません。

ハートビートの設定は、カメラオープン時に 15 秒ハートビートタイムアウト設定でハートビートが有効に設定されます。 [set_heartbeat\(\)](#) 関数で設定を変更することができます。

7.3.3. close 関数

カメラをクローズします。

[構文]

```
pytelicam.CameraDevice.close(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

別のスレッドでカメラの制御等を行っているときは、そのスレッドの処理が終了するまで本関数を実行しないでください。別のスレッドで例外エラーが発生する場合があります。

7.3.4. read_register 関数

カメラのレジスタからデータを取得します。

[構文]

```
pytelicam.CameraDevice.read_register(self, address)
```

[パラメータ]

パラメータ	内 容
address (int)	カメラレジスタの読み出しアドレス

[戻り値]

取得した値

[戻り値の型]

int

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.3.5. write_register 関数

カメラのレジスタにデータを書き込みます。

[構文]

```
pytelicam.CameraDevice.write_register(self, address, value)
```

[パラメータ]

パラメータ	内 容
address (int)	カメラレジスタの読み出しアドレス
value (int)	書き込む値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功。

CamApiStatus.NotWritable : 書き込み可能な状態ではないため、失敗しました。
カメラがストリーミング中のため、カメラがレジスタへの
書き込みを拒否している可能性があります。

CamApiStatus.NotImplemented : 機能が実装されていないレジスタにアクセスし、失敗しました。

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

7.3.6. reset_port 関数

ホストアダプタ／ホストコントローラのポートリセットを実行します。

[構文]

```
pytelicam.CameraDevice.reset_port(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数は、USB3 カメラに対してのみ有効です。

カメラがオープンされているときのみポートリセットが実行できます。

ポートリセットを実行するとカメラ取り外しイベントが発生します。

本関数を実行すると、プラグ&プレイサービスが走り、USB コントローラとカメラにはデバイスドライバが再アタッチされます。この処理には少し時間がかかるので、カメラを再オープンする前に [get_num_of_cameras\(\)](#) 関数を実行し、接続されているはずのカメラがすべて再認識されていることを確認してください。

問題が発生した場合、復旧処理を実行するためにこの関数を実行しても、すべての場合で正常に動作が復帰するとは限りません。

7.3.7. get_heartbeat 関数

カメラのハートビート設定を取得します。

[構文]

```
pytelicam.CameraDevice.get_heartbeat(self)
```

[戻り値]

tuple 型のデータ (enable, timeout)

enable : ハートビート設定値。 True のとき有効、False のとき無効です。

timeout: ハートビートタイムアウト時間。 単位は [ms] です。

[戻り値の型]

tuple(bool, int)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数は、GigE カメラのみ有効です。

7.3.8. set_heartbeat 関数

カメラのハートビート設定を変更します。

[構文]

```
pytelicam.CameraDevice.set_heartbeat(self, enable, timeout)
```

[パラメータ]

パラメータ	内 容
enable (bool)	ハートビート設定値 True のとき有効、False のとき無効です。
timeout (int)	ハートビートタイムアウト時間 (単位 : ms) 設定できる最小値は 500(ms) です。

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数は、GigE カメラ使用時のみ有効です。また、カメラによりハートビート無効に設定できないカメラがあります。（GiantDragon シリーズは、ハートビート無効に設定できません。）

GigE カメラの場合、カメラオープン時に 15 秒のハートビートタイムアウト設定でハートビートが必ず有効に設定されます。通常はハートビート設定を変更する必要はありませんが、デバッグ等で長時間処理を中断させる場合には、本関数を使用してハートビート設定を変更してください。

ハートビートの管理は pytelicam (TeliCamAPI) で一元管理しています。 [GenApiWrapper](#) クラスの関数または [write_register\(\)](#) 関数を使用してカメラの `GevHeartbeatTimeout` レジスタおよび `GevGCCPHeartbeatDisable` レジスタの値を変更しないでください。

7.3.9. save_parameter 関数

GenICam GenAPI の機能を利用して、カメラのパラメータをセーブします。

[構文]

```
pytelicam.CameraDevice.save_parameter(self, file_full_path)
```

[パラメータ]

パラメータ	内 容
file_full_path (str)	保存するファイルのフルパス

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

保存されるパラメータは、カメラ記述ファイル（XML ファイル）で定義されています。
通常、カメラ記述ファイルはカメラから読み出されます。

セーブするファイルは、フルパスで指定する必要があります。また、作成されるファイルはテキストファイル形式です。下記の設定例を参考に指定を行って下さい。

プラットフォーム	設定例
Windows の場合	"C:\\\\Work\\ParameterFile.txt"
Linux 系 OS の場合	"\$HOME/ParameterFile.txt"

7.3.10. load_parameter 関数

GenICam GenAPI の機能を利用して、カメラのパラメータをロードします。

[構文]

```
pytelicam.CameraDevice.load_parameter(self, file_full_path)
```

[パラメータ]

パラメータ	内 容
file_full_path (str)	ロードするファイルのフルパス

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

[save_parameter\(\)](#) 関数で保存したファイル以外はロードできません。

画素欠陥補正データは、個々のカメラ毎に調整されたデータです。
ファイルに保存されている画素欠陥補正データはカメラにロードされません。

7.3.11. `get_last_genicam_error` 関数

GenICamError が発生したときのエラー情報を取得します。

[構文]

```
pytelicam.CameraDevice.get_last_genicam_error(self)
```

[戻り値]

最後に発生した GenICamError のメッセージ

[戻り値の型]

str

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

本関数を実行する前または実行中に他のスレッドで GenICamError が発生した場合は、エラー情報が上書きされてしまう場合があります。

7.4. CameraStream クラス (ストリーム制御)

TeliCamAPI の高水準関数を使用したストリーム（画像）クラスです。

本クラスは、カメラから受け取った画像データをユーザーアプリケーションに渡すための [ImageData](#) オブジェクトを本 API 内部のストリームリングバッファで管理します。

ユーザーアプリケーションは本クラスの関数を使用して画像データを取り出すことができます。また、本クラスが提供するコールバック関数を使用して各イベント発生時の処理を簡単に実装することもできます。





本クラスのオブジェクトは [CameraDevice](#) クラスで管理されており、ユーザーアプリケーションで オブジェクトの作成および破棄 (del)を行う必要はありません。

[構文]


```
pytelicam.CameraStream
```

[関数]

	名 称	説 明
⇒	open	画像データ取得用のストリームインターフェースをオープンします。
⇒	close	ストリームインターフェースをクローズします。
⇒	start	画像データの転送開始をカメラに要求します。
⇒	stop	画像データの転送停止をカメラに要求します。
⇒	abort	画像データの転送中断をカメラに要求します。
⇒	get_current_buffer_index	最新の画像データを格納している本 API 内部のストリームリングバッファのバッファインデックスを取得します。
⇒	get_next_image	次に到着する画像データが本 API 内部のストリームリングバッファに格納されるまで待機し、格納処理が完了したら画像データを取得します。
⇒	get_next_image_with_trigger	ソフトウェアトリガコマンドを発行し、次に到着する画像データを取得します。
⇒	get_current_buffered_image	本 API 内部のストリームリングバッファから、格納済の最新画像データを取得します。
⇒	get_buffered_image	本 API 内部のストリームリングバッファに格納された画像データを取得します。 バッファインデックスにより、バッファの位置を指定します。
⇒	chunk_attach_buffer	GenICam GenApi のチャンクアダプタに、バッファをアタッチします。
⇒	chunk_update_buffer	GenICam GenApi のチャンクアダプタにアタッチされているバッファを更新します。
⇒	chunk_check_buffer_layout	指定されたバッファに既知の形式のチャンクデータが含まれているかどうか確認します。
⇒	set_callback_image_acquired	新しい画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数を登録します。
⇒	reset_callback_image_acquired	新しい画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数の登録を解除します。

	set_callback_image_error	異常な画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数を登録します。
	reset_callback_image_error	異常な画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数の登録を解除します。
	set_callback_buffer_busy	バッファービジーエラーが発生したときに呼び出すコールバック関数を登録します。
	reset_callback_buffer_busy	バッファービジーエラーが発生したときに呼び出すコールバック関数の登録を解除します。

[プロパティ]

	名 称	内 容
	is_open (bool)	ストリームインターフェースのオープン/クローズ状態を示す値。 True のときオープン状態、False のときクローズ状態です。

7.4.1. open 関数

画像データ取得用のストリームインターフェースをオープンします。

本関数は、デバイスドライバから受け取った画像データをユーザーアプリケーションに渡すための [ImageData](#) オブジェクトを作成し、本 API 内部のストリームリングバッファで管理します。

【構文】

```
pytelicam.CameraStream.open(  
    self, acquired_signal=None, api_buffer_count=0, max_packet_size=0)
```

【パラメータ】

パラメータ	内 容														
acquired_signal (pytelicam.SignalHandle)	画像データ受信を通知するためのシグナルオブジェクト。 新しい画像データの受信を完了すると、シグナル状態に変化します。 画像データの受信を知る必要がない時は <code>None</code> を指定します。														
api_buffer_count (int)	本API内部で管理するストリームリングバッファの数。 1～128 の値を指定できます。 0を指定すると、ペイロードサイズに応じて以下の値が指定されたことになります。 <table><tr><th>ペイロードサイズ</th><th>バッファ数</th></tr><tr><td>16MByte未満</td><td>8</td></tr><tr><td>16MByte以上 24MByte未満</td><td>7</td></tr><tr><td>24MByte以上 32MByte未満</td><td>6</td></tr><tr><td>32MByte以上 40MByte未満</td><td>5</td></tr><tr><td>40MByte以上 48MByte未満</td><td>4</td></tr><tr><td>48MByte以上</td><td>3</td></tr></table>	ペイロードサイズ	バッファ数	16MByte未満	8	16MByte以上 24MByte未満	7	24MByte以上 32MByte未満	6	32MByte以上 40MByte未満	5	40MByte以上 48MByte未満	4	48MByte以上	3
ペイロードサイズ	バッファ数														
16MByte未満	8														
16MByte以上 24MByte未満	7														
24MByte以上 32MByte未満	6														
32MByte以上 40MByte未満	5														
40MByte以上 48MByte未満	4														
48MByte以上	3														
max_packet_size (int)	ドライバが受け取るパケットの最大サイズ。(単位: byte。省略可能) 0 を指定すると、以下のデフォルト値が指定されたことになります。 USB3 カメラ使用時 : 65536 byte GigE カメラ使用時 : ジャンボフレーム (MTU) 設定値から算出された値 GigEカメラでストリーミングのオーバーヘッド低減、スループット向上を目的としてジャンボフレームを設定する場合は、イーサネットヘッダを除いたパケットサイズ値を指定してください。 ネットワークアダプタがイーサネットヘッダを含んだジャンボフレーム値を公表している場合は、イーサネットヘッダサイズ (14byte) を減算した値を指定する必要があります。設定する値は4の倍数である必要があります。 ジャンボフレームの値が「9014」と公表されている場合、通常は、イーサネットヘッダを含んだ値が表示されています。この場合、「9000」を本パラメータに指定してください。 特別な理由がない限り 0 を指定することをお勧めします。														

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

他のアプリケーションが同じカメラのストリームインターフェースを使用中のとき、本関数はオープンに失敗し、例外エラーを出力します。

CameraStream オブジェクトは新しい画像データをカメラから受け取り、ストリームリングバッファへの保存が完了すると、[acquired signal](#) をシグナル状態にセットし、[set_callback_image_acquired\(\)](#) 関数で登録されたコールバック関数を実行します。

CameraStream クラスはストリームリングバッファ内の画像データを取得する手段を3種類提供しています。

1. [get_next_image\(\)](#) 関数を使用

次に到着する画像データを取得します。

2. [get_current_buffered_image\(\)](#) 関数を使用

本 API 内部のストリームリングバッファから、格納済の最新画像データを取得します。

3. [get_current_buffer_index\(\)](#)、[get_buffered_image\(\)](#) 関数を使用

この方法は、ストリームリングバッファ内にある任意のインデックスの画像データを取得できます。以下の手順で画像データを取得してください。

- A. [get_current_buffer_index\(\)](#) 関数で、最新画像データを格納しているストリームリングバッファのインデックスを取得。
- B. 取得したい画像データのインデックスを計算。
- C. [get_buffered_image\(\)](#) 関数で、取得したい画像データを格納している [ImageData](#) オブジェクトを取得し、画像データを取り出します。
- D. 画像データの取り出しが完了したら、[release\(\)](#) 関数により [ImageData](#) オブジェクトを開放し、[ImageData](#) オブジェクトをストリームリングバッファに戻します。

優先度が高いスレッドが並行動作していると、[acquired signal](#) がシグナル状態になったとき、またはコールバック関数が実行された時には複数の画像が受信されている場合があります。このとき、[acquired signal](#) のシグナル状態へのセットおよびコールバック関数の実行は一度しか行われませんのでご注意ください。

受信したすべての画像データを必要とする場合は、前述の3. の手順ですべての画像データをストリームリングバッファから取り出してください。

本 API 内部のストリームリングバッファは、カメラ（内部）のイメージバッファとは異なりますのでご注意ください。

7.4.2. close 関数

画像データ取得用のストリームインターフェースをクローズします。

[構文]

```
pytelicam.CameraStream.close(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

ストリームインターフェースをオープンした場合は、必ず本関数によりストリームインターフェースをクローズしてください。

別のスレッドで `CameraStream` オブジェクトを使用する処理を行っているときに本関数を実行すると他のスレッドでエラーが発生します。すべてのスレッドで `CameraStream` オブジェクトを使用する処理が終了した後に本関数を実行してください。

7.4.3. start 関数

画像データの転送開始をカメラに要求します。

[構文]

```
pytelicam.CameraStream.start(  
    self,  
    acquisition_mode=pytelicam.CameraAcquisitionMode.Continuous)
```

[パラメータ]

パラメータ	内 容
acquisition_mode (pytelicam.CameraAcquisitionMode)	画像データ転送モード

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

カメライメージバッファ転送モード([CameraAcquisitionMode](#). ImageBufferRead)で使用する場合は、カメラの ImageBufferMode レジスタの値を On に、それ以外のモードで使用する場合は Off に設定してください。

7.4.4. stop 関数

画像データの転送停止をカメラに要求します。

[構文]

```
pytelicam.CameraStream.stop(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.4.5. abort 関数

画像データの転送中断をカメラに要求します。

[構文]

```
pytelicam.CameraStream.abort(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

中断されたフレームにはエラーのステータスが付与されます。

本関数により画像データの転送が中断された場合は、[set_callback_image_error\(\)](#) 関数で設定したコールバック関数はコールされません。

7.4.6. get_current_buffer_index 関数

最新の画像データを格納している本 API 内部のストリームリングバッファのバッファインデックスを取得します。

[構文]

```
pytelicam.CameraStream.get_current_buffer_index(self)
```

[戻り値]

バッファインデックス

[戻り値の型]

int

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

バッファインデックスの値は、0 から CameraStream オブジェクトの [open\(\)](#) 関数の [api_buffer_count](#) 引数で指定した値-1 です。

ただし、画像が一枚も取り込まれていない場合は -1 がリターンされます。

7.4.7. get_next_image 関数

次に到着する画像データが本 API 内部のストリームリングバッファに格納されるまで待機し、格納処理が完了したら画像データを取得します。

[構文]

```
pytelicam.CameraStream.get_next_image(self, timeout=5000)
```

[パラメータ]

パラメータ	内 容
timeout (int)	次の画像データが到着するまで待機するタイムアウト時間。(単位：ミリ秒、デフォルト値：5000) 0 を指定すると、待機せずすぐに結果を返します。 -1 または 0xFFFFFFFF を指定すると、次の画像が到着するまで待機し続けます。

[戻り値]

画像データのオブジェクト

[戻り値の型]

[pytelicam.ImageData](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

タイムアウトエラーが発生、または異常な画像データがバッファに格納されている場合があります。異常な画像データとは、以下を示します。

- ・カメラが転送する画像データの付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常に画像データを取り込めなかった場合

戻り値 [ImageData](#) オブジェクトのステータス (status) を確認することで正常な画像データが格納されているかどうかを知ることができます。

戻り値の [ImageData](#) オブジェクトはロックされています。 開放せずに画像データの取り込みを繰り返した場合、本 API 内部のストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、画像データを取得することができなくなります。 使用後は速やかに開放してください。

開放する方法は、[release\(\)](#) 関数を呼び出すか、python の with 構文を使用します。

```
image_data = cam_device.cam_stream.get_next_image()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

または

```
with cam_device.cam_stream.get_next_image() as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```


7.4.8. get_next_image_with_trigger 関数

ソフトウェアトリガコマンドを発行し、次に到着する画像データを取得します。

カメラのトリガモードを ON、トリガソースをソフトウェアトリガに設定する必要があります。

[構文]

```
pytelicam.CameraStream.get_next_image_with_trigger(self, timeout=5000)
```

[パラメータ]

パラメータ	内 容
timeout (int)	次の画像データが到着するまで待機するタイムアウト時間。(単位：ミリ秒、デフォルト値：5000) 0 を指定すると、待機せずすぐに結果を返します。 -1 または 0xFFFFFFFF を指定すると、次の画像が到着するまで待機し続けます。

[戻り値]

画像データのオブジェクト

[戻り値の型]

[pytelicam.ImageData](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

タイムアウトエラーが発生、または異常な画像データが取得される場合があります。

異常な画像データとは、以下を示します。

- ・カメラが転送する画像データの付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常に画像データを取り込めなかった場合

戻り値 [ImageData](#) オブジェクトのステータス (status) を確認することで正常な画像データが格納されているかどうかを知ることができます。

戻り値の [ImageData](#) オブジェクトはロックされています。 開放せずに画像データの取り込みを繰り返した場合、本 API 内部のストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、画像データを取得することができなくなります。 使用後は速やかに開放してください。

開放する方法は、[release\(\)](#) 関数を呼び出すか、python の with 構文を使用します。

```
cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
cam_device.genapi.set_enum_str_value('TriggerSource', 'Software')

image_data = cam_device.cam_stream.get_next_image_with_trigger()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {}'.format(image_data.status))

image_data.release()
```

または

```
cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
cam_device.genapi.set_enum_str_value('TriggerSource', 'Software')

with cam_device.cam_stream.get_next_image_with_trigger() as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {}'.format(image_data.status))
```

7.4.9. `get_current_buffered_image` 関数

本 API 内部のストリームリングバッファから、格納済の最新画像データを取得します。

[構文]

```
pytelicam.CameraStream.get_current_buffered_image(self)
```

[戻り値]

画像データのオブジェクト

[戻り値の型]

[pytelicam.ImageData](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

異常な画像データがバッファに格納されている場合があります。

異常な画像データとは、以下を示します。

- ・カメラが転送する画像データの付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常に画像データを取り込めなかった場合

戻り値 [ImageData](#) オブジェクトのステータス (status) を確認することで正常な画像データが格納されているかどうかを知ることができます。

戻り値の [ImageData](#) オブジェクトはロックされています。 開放せずに画像データの取り込みを繰り返した場合、本 API 内部のストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、画像データを取得することができなくなります。 使用後は速やかに開放してください。

開放する方法は、[release\(\)](#) 関数を呼び出すか、python の with 構文を使用します。

```
image_data = cam_device.cam_stream.get_current_buffered_image()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

または

```
with cam_device.cam_stream.get_current_buffered_image() as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```

7.4.10. get_buffered_image 関数

本 API 内部のストリームリングバッファに格納された画像データを取得します。
バッファインデックスにより、バッファの位置を指定します。

[構文]

```
pytelicam.CameraStream.get_buffered_image(self, buffer_index)
```

[パラメータ]

パラメータ	内 容
buffer_index (int)	ストリームリングバッファのバッファインデックス。 指定できる値は、0 から CameraStream オブジェクトの open() 関数の api_buffer_count 引数で指定した値-1 です。

[戻り値]

画像データのオブジェクト

[戻り値の型]

[pytelicam.ImageData](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

異常な画像データがバッファに格納されている場合があります。

異常な画像データとは、以下を示します。

- ・カメラが転送する画像データの付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常に画像データを取り込めなかった場合

戻り値 [ImageData](#) オブジェクトのステータス (status) を確認することで正常な画像データが格納されているかどうかを知ることができます。

戻り値の [ImageData](#) オブジェクトはロックされています。 開放せずに画像データの取り込みを繰り返した場合、本 API 内部のストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、画像データを取得することができなくなります。 使用後は速やかに開放してください。

開放する方法は、[release\(\)](#) 関数を呼び出すか、python の with 構文を使用します。

```
image_data = cam_device.cam_stream.get_buffered_image(0)
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

または

```
with cam_device.cam_stream.get_buffered_image(0) as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```

7.4.11. chunk_attach_buffer 関数

GenICam GenApi のチャンクアダプタに、バッファをアタッチします。

GenICam GenApi を使用してチャンクデータを取得するとき、[ImageData](#) オブジェクトが管理しているバッファを GenICam GenApi のチャンクアダプタにアタッチする必要があります。

GenTL インターフェース（CoaXPress カメラ）には対応していません。

[構文]

```
pytelicam.CameraStream.chunk_attach_buffer(self, image_data)
```

[パラメータ]

パラメータ	内 容
image_data (pytelicam.ImageData)	GenICam GenApi のチャンクアダプタにアタッチする ImageData オブジェクト

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

画像データにチャンクデータが付加されていない場合は例外が送出されます。

チャンクデータは、[GenApiWrapper](#) クラスを使用して取得します。

```
image_data = cam_device.cam_stream.get_current_buffered_image()
cam_device.cam_stream.chunk_attach_buffer(image_data)

res, frame_id = cam_device.genapi.get_int_value('ChunkFrameID')
res, exposure_time = cam_device.genapi.get_float_value('ChunkExposureTime')
res, gain = cam_device.genapi.get_float_value('ChunkGain')

print('  ChunkFrameID      : ', frame_id)
print('  ChunkExposureTime : ', exposure_time)
print('  ChunkGain          : ', gain)

image_data.release()
```

7.4.12. chunk_update_buffer 関数

GenICam GenApi のチャンクアダプタにアタッチされているバッファを更新します。
GenTL インターフェース（CoaXPress カメラ）には対応していません。

[構文]

```
pytelicam.CameraStream.chunk_attach_buffer(self, image_data)
```

[パラメータ]

パラメータ	内 容
image_data (pytelicam.ImageData)	GenICam GenApi のチャンクアダプタにアタッチする ImageData オブジェクト

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

画像データにチャンクデータが付加されていない場合は例外が送出されます。
チャンクデータは、[GenApiWrapper](#) クラスを使用して取得します。

GenICam GenApi のチャンクアダプタにバッファが一度もアタッチされていない場合、またはチャンクデータのレイアウトが変更されている場合は例外エラーが出力されます。

[chunk_attach_buffer\(\)](#) 関数より高速に処理できます。

7.4.13. chunk_check_buffer_layout 関数

指定されたバッファに既知の形式のチャンクデータが含まれているかどうか確認します。

GenTL インターフェース（CoaXPress カメラ）には対応していません。

[構文]

```
pytelicam.CameraStream.chunk_check_buffer_layout(self, image_data)
```

[パラメータ]

パラメータ	内 容
image_data (pytelicam.ImageData)	確認するバッファを管理している ImageData オブジェクト

[戻り値]

実行結果。

True のとき、有効なチャンクデータが含まれています。

False のとき、有効なチャンクデータが含まれていません。

[戻り値の型]

bool

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.4.14. set_callback_image_acquired 関数

新しい画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数を登録します。

[構文]

```
pytelicam.CameraStream.set_callback_image_acquired(self, func)
```

[パラメータ]

パラメータ	内 容
func	登録するコールバック関数

func で登録される関数は、次の形式である必要があります。

```
func(image_data)
```

パラメータ	内 容
image_data (pytelicam.ImageData)	画像データのオブジェクト

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

登録できる関数は一つです。

image_data で管理されているバッファはロックされています。 コールバック関数の処理はできるだけ短くしてください。

```
def callback_image_acquired(image_data):
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))

cam_system = pytelicam.get_camera_system()
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open()
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.start()
...
```

7.4.15. reset_callback_image_acquired 関数

新しい画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数の登録を解除します。

[構文]

```
pytelicam.CameraStream.reset_callback_image_acquired(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.4.16. set_callback_image_error 関数

異常な画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数を登録します。

[構文]

```
pytelicam.CameraStream.set_callback_image_error(self)
```

[パラメータ]

パラメータ	内 容
func	登録するコールバック関数

func で登録される関数は、次の形式である必要があります。

```
func(status, buffer_index)
```

パラメータ	内 容
status (pytelicam.CamApiStatus)	エラーのステータスコード
buffer_index (int)	異常な画像データ (ImageData オブジェクト) を格納している本 API 内部のストリームリングバッファのバッファインデックス

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

登録できる関数は一つです。

異常な画像データとは、以下を示します。

- ・カメラが転送する画像データの付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常に画像データを取り込めなかった場合

エラーの内容は、status で知ることができます。

```
def callback_image_error(status, buffer_index):
    print('ImageError Callback! , status={0}, buffer_index={1}' \
          .format(status, buffer_index))

cam_system = pytelicam.get_camera_system()
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open()
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.set_callback_image_error(callback_image_error)
...
```

7.4.17. reset_callback_image_error 関数

異常な画像データが本 API 内部のストリームリングバッファに格納されたときに呼び出すコールバック関数の登録を解除します。

[構文]

```
pytelicam.CameraStream.reset_callback_image_error(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.4.18. set_callback_buffer_busy 関数

バッファービジーエラーが発生したときに呼び出すコールバック関数を登録します。

[構文]

```
pytelicam.CameraStream.set_callback_buffer_busy(self, func)
```

[パラメータ]

パラメータ	内 容
func	登録するコールバック関数

func で登録される関数は、次の形式である必要があります。

```
func(buffer_index)
```

パラメータ	内 容
buffer_index (int)	ロックされていてエラーを発生させた本 API 内部のストリームリングバッファのバッファインデックス

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

登録できる関数は一つです。

バッファービジーエラーは、本 API 内部のストリームリングバッファの書き込み先バッファがアプリケーションによりロックされていて、取得した画像がバッファに格納されずに破棄されたことを示します。

バッファービジーエラーが発生する場合は、[ImageData](#) オブジェクトをロックしている期間を短縮するか、バッファの数を増やすことで改善する場合があります。 バッファの数は、[CameraStream](#) オブジェクトの [open\(\)](#) 関数の [api_buffer_count](#) で設定します。

```
def callback_buffer_busy(buffer_index):
    print('BufferBusy Callback! , buffer_index={0}'.format(buffer_index))

cam_system = pytelicam.get_camera_system()
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open(None, 32)
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.set_callback_buffer_busy(callback_buffer_busy)
...
```

7.4.19. reset_callback_buffer_busy 関数

バッファービジーエラーが発生したときに呼び出すコールバック関数の登録を解除します。

【構文】

```
pytelicam.CameraStream.reset_callback_buffer_busy(self)
```

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.5. CameraEvent クラス (イベント制御)

TeliCamAPI の低水準関数を使用したカメライベント通知（メッセージ）を管理するクラスです。

本クラスは、デバイスドライバから受け取ったカメライベント通知（メッセージ）をユーザーアプリケーションに渡すための [EventData](#) オブジェクトを本 API 内部のイベント FIFO バッファで管理します。

ユーザーアプリケーションは本オブジェクトの関数を使用してカメライベント通知（メッセージ）を受け取ることができます。

受信したカメライベント通知（メッセージ）のデータは、GenICam GenApi に配信されません。そのため、Timestamp 等のイベント情報は [GenApiWrapper](#) オブジェクトの関数を使用して取得することはできません。

CameraEvent クラスのオブジェクトは [CameraDevice](#) クラスで管理されており、ユーザーアプリケーションで オブジェクトの作成および破棄 (del)を行う必要はありません。


[構文]

```
pytelicam.CameraEvent
```

[関数]

	名 称	説 明
	open	カメライベント通知（メッセージ）取得用のイベントインターフェースをオープンします。
	close	イベントインターフェースをクローズします。
	activate	カメライベント通知（メッセージ）を有効にします。
	deactivate	カメライベント通知（メッセージ）を無効にします。
	get_event_data	本 API 内部のイベント FIFO バッファから、格納済のカメライベントデータを取り出します。
	clear_event_data	本 API 内部のイベント FIFO バッファに格納済のカメライベントデータをクリアします。
	get_queueing_count	本 API 内部のイベント FIFO バッファに格納されているカメライベントデータの数を取得します。
	get_lost_count	本 API 内部のイベント FIFO バッファに空きがなかったために破棄されたカメライベントデータの数を取得します。

[プロパティ]

	名 称	内 容
	is_open (bool)	イベントインターフェースのオープン/クローズ状態を示す値。 True のときオープン状態、False のときクローズ状態です。

7.5.1. open 関数

カメライベント通知（メッセージ）取得用のイベントインターフェースをオープンします。

本関数は、デバイスドライバから受け取ったカメライベント通知（メッセージ）をユーザーアプリケーションに渡すためのバッファとして、pytelicam 内部に [EventData](#) オブジェクトを作成し、pytelicam 内部のイベント FIFO バッファで管理します。

【構文】

```
pytelicam.CameraEvent.open(self, api_buffer_count=32)
```

【パラメータ】

パラメータ	内 容
api_buffer_count (int)	本API内部で管理するイベントFIFOバッファのサイズ。 1～128 の値が指定できます。

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

【備考】

他のアプリケーションが同じカメラのイベントインターフェースを使用中のとき、本関数はオープンに失敗し、例外エラーを出力します。

カメライベント通知（メッセージ）は、[get_event_data\(\)](#) 関数でカメラから受信した順番に取得します。

```
cam_device.cam_event.open()
cam_device.cam_event.activate(pytelicam.CameraEventType.ExposureEnd)

acquired_signal = cam_system.create_signal()
cam_device.cam_stream.open(acquired_signal)
cam_device.cam_stream.start()

for i in range(10):
    res = cam_device.genapi.execute_command('TriggerSoftware')
    if res != pytelicam.CamApiStatus.Success:
        print("Can't execute TriggerSoftware.")
        break

# If you don't use 'with' statement, event_data.release() must be
# called after using event_data.
with cam_device.cam_event.get_event_data(1000) as event_data :
    if event_data.status == pytelicam.CamApiStatus.Success:
        print('\nrequest_id : {0}, event_id : {1}, timestamp : {2}, ' \
              .format( \
                  hex(event_data.request_id), \
                  hex(event_data.event_id), \
                  event_data.timestamp))
```

```
        else:
            print('Error : event_data was not received.')

    res = cam_system.wait_for_signal(acquired_signal)
    if res != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(res))
        break
    else:
        print('Image Acquired.')

cam_device.cam_stream.stop()
cam_device.cam_stream.close()
cam_device.cam_event.deactivate(pytelicam.CameraEventType.ExposureEnd)
cam_device.cam_event.close()
```

1つのカメラに複数のカメライベント通知を設定した場合、複数のカメライベント通知が短時間に受信される場合があります。 使用環境やアプリケーションにより処理が間に合わなくなった場合、処理しきれなくなったカメライベント通知は本 API 内部のイベント FIFO バッファに格納されることなく破棄されます。

カメライベント通知を取り漏らす場合は、`api_buffer_count` の数を増やすことで改善する場合があります。 また、[get_event_data\(\)](#) 関数で取得した [EventData](#) オブジェクトは、使用が完了したら速やかに開放してください。

7.5.2. close 関数

カメライベント通知（メッセージ）取得用のイベントインターフェースをクローズします。

[構文]

```
pytelicam.CameraEvent.close(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

イベントインターフェースをオープンした場合は、必ず本関数によりイベントインターフェースをクローズしてください。

別のスレッドで CameraEvent オブジェクトを使用する処理を行っているときに本関数を実行すると他のスレッドでエラーが発生します。すべてのスレッドで CameraEvent オブジェクトを使用する処理が終了した後に本関数を実行してください。

7.5.3. activate 関数

指定されたカメライベント通知（メッセージ）を有効にします。

[構文]

```
pytelicam.CameraEvent.activate(self, event_type)
```

[パラメータ]

パラメータ	内 容
event_type (pytelicam.CameraEventType)	有効にするカメライベント通知（メッセージ）の種類

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

この関数は、カメラの EventSelector ノードと EventNotification ノードの設定を行います。

[CameraEventType](#) で定義されていないカメライベント通知（メッセージ）を有効化するには、[GenApiWrapper](#) オブジェクトを使用して設定します。

カメラに設定する機能の順番により、例外が送出される場合があります。これは、カメラのモデルまたはファームウェアバージョンに依存します。

例えば、一部の GigE カメラでは、FrameTrigger イベントは TriggerMode = On の時にしか設定できません。

カメラのモデルによって通知出来るイベントの種類は異なりますので、詳細についてはカメラの取扱説明書をご覧ください。

7.5.4. deactivate 関数

指定されたカメライベント通知（メッセージ）を無効にします。

【構文】

```
pytelicam.CameraEvent.deactivate(self, event_type)
```

【パラメータ】

パラメータ	内 容
event_type (pytelicam.CameraEventType)	無効にするカメライベント通知（メッセージ）の種類

【戻り値】

なし

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.5.5. get_event_data 関数

本 API 内部のイベント FIFO バッファから、格納済のカメライベントデータを取り出します。
カメライベントデータは、格納された順番に一つずつ取り出されます。

[構文]

```
pytelicam.CameraEvent.get_event_data(self, timeout=0)
```

[パラメータ]

パラメータ	内 容
timeout (int)	取り出されていないカメライベントデータが本 API 内部のイベント FIFO バッファに存在しない場合、新しいカメライベント通知（メッセージ）が到着するまで待機するタイムアウト時間。（単位：ミリ秒） 0 を指定すると、待機せずすぐに結果を返します。 0xFFFFFFFF を指定すると、次のカメライベント通知（メッセージ）が到着するまで待機します。

[戻り値]

イベントデータのオブジェクト

[戻り値の型]

[pytelicam.EventData](#)

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

タイムアウトエラーが発生、または異常なカメライベントデータがバッファに格納されている場合があります。

異常なカメライベントデータとは、以下を示します。

- ・カメラが転送するカメライベント通知（メッセージ）の付随情報にエラーを設定されている場合
- ・パケット損失等が発生し、正常にカメライベント通知（メッセージ）を取り込めなかった場合

戻り値 [EventData](#) オブジェクトのステータス（status）を確認することで正常なカメライベントデータが格納されているかどうかを知ることができます。

戻り値の [EventData](#) オブジェクトはロックされています。 開放せずに画カメライベント通知（メッセージ）の取り込みを繰り返した場合、本 API 内部のイベント FIFO バッファに未使用の [EventData](#) オブジェクトがなくなり、カメライベント通知（メッセージ）を取得することができなくなります。使用後は速やかに開放してください。

開放する方法は、[release\(\)](#) 関数を呼び出すか、python の with 構文を使用します。

7.5.6. clear_event_data 関数

本 API 内部のイベント FIFO バッファに格納済のカメライベントデータをクリアします。

[構文]

```
pytelicam.CameraEvent.clear_event_data(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.5.7. get_queueing_count 関数

本 API 内部のイベント FIFO バッファに格納されているカメライベントデータの数を取得します。

[構文]

```
pytelicam.CameraEvent.get_queueing_count(self)
```

[戻り値]

イベント FIFO バッファに格納済のカメライベントデータの数

[戻り値の型]

int

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.5.8. get_lost_count 関数

本 API 内部のイベント FIFO バッファに空きバッファがなくなったため破棄されたカメライベント通知（メッセージ）の数を取得します。

[構文]

```
pytelicam.CameraEvent.get_lost_count(self)
```

[戻り値]

損失したイベントデータの数

[戻り値の型]

int

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

[get_event_data\(\)](#) 関数で取得した [EventData](#) オブジェクトは、使用が完了したら速やかに開放してください

7.6. CameraControl クラス

カメラの制御を行うクラスです。

カメラのインターフェース、モデル、レジスタアドレスなどを意識することなく、カメラを簡単に制御することができます。

IIDC2 規格に準拠しているカメラでは、処理パフォーマンスを高めるために GenICam GenApi ライブラリを使用せず、レジスタアクセスで処理されます。（一部のメソッドを除く）

IIDC2 規格に準拠していないカメラでは、GenICam GenApi ライブラリが使用されます。カメラオープン時に GenApi モジュール無効を指定した場合は、本章のメソッドは使用できなくなりますのでご注意ください。

カメラにより、利用できるメソッドが異なります。機能が実装されているかどうかは、使用するカメラの取扱説明書をご覧ください。

CameraControl クラスのオブジェクトは [CameraDevice](#) クラスで管理されており、ユーザーアプリケーションで オブジェクトの作成および破棄 (del)を行う必要はありません。

[構文]

pytelicam.CameraControl

[関数]

	名 称	説 明
ImageFormatControl		
⇒	get_image_format_selector	カメラの映像フォーマットを取得します。
⇒	set_image_format_selector	カメラの映像フォーマットを設定します。
Scalable		
⇒	get_sensor_width	カメラの水平有効画素数を取得します。
⇒	get_sensor_height	カメラの垂直有効画素数を取得します。
⇒	get_roi	カメラの ROI（領域）を取得します。
⇒	set_roi	カメラの ROI（領域）を設定します。
⇒	get_width_min_max	カメラに設定できる映像の幅の最小値・最大値・増加量を取得します。
⇒	get_width	カメラの映像の幅を取得します。
⇒	set_width	カメラの映像の幅を設定します。
⇒	get_height_min_max	カメラに設定できる映像の高さの最小値・最大値・増加量を取得します。
⇒	get_height	カメラの映像の高さを取得します。
⇒	set_height	カメラの映像の高さを設定します。
⇒	get_offset_x_min_max	カメラに設定できる映像の水平方向開始位置の最小値・最大値・増加量を取得します。
⇒	get_offset_x	カメラの映像の水平方向開始位置を取得します。
⇒	set_offset_x	カメラの映像の水平方向開始位置を設定します。
⇒	get_offset_y_min_max	カメラに設定できる映像の垂直方向開始位置の最小値・最大値・増加量を取得します。
⇒	get_offset_y	カメラの映像の垂直方向開始位置を取得します。

	名 称	説 明
⇒	set_offset_y	カメラの映像の垂直方向開始位置を設定します。
Binning		
⇒	get_binning_horizontal_min_max	カメラに設定できる水平方向ビニング設定値の最小値と最大値を取得します。
⇒	get_binning_horizontal	カメラの水平方向ビニング設定値を取得します。
⇒	set_binning_horizontal	カメラの水平方向ビニングを設定します。
⇒	get_binning_vertical_min_max	カメラに設定できる垂直方向ビニング設定値の最小値と最大値を取得します。
⇒	get_binning_vertical	カメラの垂直方向ビニング設定値を取得します。
⇒	set_binning_vertical	カメラの垂直方向ビニングを設定します。
Decimation		
⇒	get_decimation_horizontal_min_max	カメラに設定できる水平方向デシメーション（間引き）設定値の最小値と最大値を取得します。
⇒	get_decimation_horizontal	カメラの水平方向デシメーション（間引き）設定値を取得します。
⇒	set_decimation_horizontal	カメラの水平方向デシメーション（間引き）を設定します。
⇒	get_decimation_vertical_min_max	カメラに設定できる垂直方向デシメーション（間引き）設定値の最小値と最大値を取得します。
⇒	get_decimation_vertical	カメラの垂直方向デシメーション（間引き）設定値を取得します。
⇒	set_decimation_vertical	カメラの垂直方向デシメーション（間引き）を設定します。
Reverse		
⇒	get_reverse_x	カメラの水平方向の映像反転設定値を取得します。
⇒	set_reverse_x	カメラの水平方向の映像反転 ON/OFF を設定します。
⇒	get_reverse_y	カメラの垂直方向の映像反転設定値を取得します。
⇒	set_reverse_y	カメラの垂直方向の映像反転 ON/OFF を設定します。
PixelFormat		
⇒	get_pixel_format	カメラの映像ストリームのピクセルフォーマット設定値を取得します。
⇒	set_pixel_format	カメラの映像ストリームのピクセルフォーマットを設定します。
TestPattern		
⇒	get_test_pattern	カメラのテストパターン設定値を取得します。
⇒	set_test_pattern	カメラのテストパターンを設定します。
AcquisitionControl		
⇒	get_stream_payload_size	カメラのレジスタから、映像ストリーム ペイロードサイズ（画像サイズ）を取得します。
⇒	get_stream_enable	カメラのレジスタから、ストリームの状態を取得します。
⇒	get_acquisition_frame_count_min_max	“AcquisitionFrameCount” の最小値と最大値を取得します。
⇒	get_acquisition_frame_count	“AcquisitionFrameCount” の値を取得します。
⇒	set_acquisition_frame_count	“AcquisitionFrameCount” の値を設定します。
⇒	get_acquisition_frame_rate_control	映像のフレームレート設定を取得します。

	名 称	説 明
⇒	set_acquisition_frame_rate_control	映像のフレームレート設定を設定します。
⇒	get_acquisition_frame_rate_min_max	映像のフレームレート最小値と最大値を取得します。
⇒	get_acquisition_frame_rate	映像のフレームレート設定値を取得します。
⇒	set_acquisition_frame_rate	映像のフレームレートを設定します。
⇒	execute_acquisition_start	AcquisitionStart コマンドを実行します。
⇒	get_high_framerate_mode	カメラの高フレームレートモードを取得します。
⇒	set_high_framerate_mode	カメラの高フレームレートモードを設定します。
ImageBuffer		
⇒	get_image_buffer_mode	カメラのイメージバッファモード設定値を取得します。
⇒	set_image_buffer_mode	カメラのイメージバッファモード ON/OFF を設定します。
⇒	get_image_buffer_frame_count	カメラのイメージバッファに取り込まれた画像枚数を取得します。
⇒	execute_image_buffer_read	カメラのイメージバッファの画像を読み出します。
TriggerControl		
⇒	get_trigger_mode	カメラのトリガ動作モードを取得します。
⇒	set_trigger_mode	カメラのトリガ動作モードを設定します。
⇒	get_trigger_sequence	カメラのトリガ ON 時における、露光時間制御モード設定値を取得します。
⇒	set_trigger_sequence	カメラのトリガ ON 時における、露光時間制御モードを設定します。
⇒	get_trigger_source	カメラのランダムトリガシャッタのトリガソース設定値を取得します。
⇒	set_trigger_source	カメラのランダムトリガシャッタのトリガソースを設定します。
⇒	get_trigger_additional_parameter_min_max	Bulk モード動作設定時の露光回数の最小値と最大値を取得します。
⇒	get_trigger_additional_parameter	Bulk モード動作設定時の露光回数を取得します。
⇒	set_trigger_additional_parameter	Bulk モード動作設定時の露光回数を設定します。
⇒	get_trigger_delay_min_max	カメラのトリガ信号検出から露光開始までの遅延量の最小値と最大値を取得します。
⇒	get_trigger_delay	カメラのトリガ信号検出から露光開始までの遅延量を取得します。
⇒	set_trigger_delay	カメラのトリガ信号検出から露光開始までの遅延量を設定します。
⇒	execute_software_trigger	カメラのソフトウェアトリガを実行します。
⇒	get_trigger_activation	カメラのソフトウェアトリガを実行します。
⇒	set_trigger_activation	カメラのソフトウェアトリガを実行します。
ExposureTime		
⇒	get_exposure_time_control	カメラの露光時間 制御モードの設定値を取得します。
⇒	set_exposure_time_control	カメラの露光時間 制御モードを設定します。
⇒	get_exposure_time_min_max	カメラの露光時間制御モードが Manual に設定されているときの、露光時間 最小値と最大値を取得します。
⇒	get_exposure_time	カメラのハードウェアトリガの有効エッジを取得します。

	名 称	説 明
⇒	set_exposure_time	カメラのハードウェアトリガの有効エッジを設定します。
⇒	get_short_exposure_mode	カメラの超短時間露光モードを取得します。
⇒	set_short_exposure_mode	カメラの超短時間露光モードを設定します。
DigitalloControl		
⇒	get_line_mode_all	カメラの全ラインに対する入出力設定値を取得します。
⇒	set_line_mode_all	カメラの全ラインに対する入出力を設定します。
⇒	get_line_mode	カメラのラインの入出力を取得します。
⇒	set_line_mode	カメラのラインの入出力を設定します。
⇒	get_line_inverter_all	カメラの全ラインに対する極性設定値を取得します。
⇒	set_line_inverter_all	カメラの全ラインに対する極性を設定します。
⇒	get_line_inverter	カメラのラインの極性を取得します。
⇒	set_line_inverter	カメラのラインの極性を設定します。
⇒	get_line_status_all	カメラの全ラインに対する現在の状態を取得します。
⇒	get_line_status	カメラのラインの現在の状態を取得します。
⇒	get_user_output_value_all	カメラの全ラインに対するユーザー出力設定値を取得します。
⇒	set_user_output_value_all	カメラの全ラインに対するユーザー出力値を設定します。
⇒	get_user_output_value	カメラのラインのユーザー出力設定値を取得します。
⇒	set_user_output_value	カメラのラインのユーザー出力設定値を設定します。
⇒	get_line_source	カメラのラインの信号種類を取得します。
⇒	set_line_source	カメラのラインの信号種類を設定します。
AntiGlitch / AntiChattering		
⇒	get_anti_glitch_min_max	カメラのデジタル入力信号の積分時間（絶対値）の最小値と最大値を取得します。
⇒	get_anti_glitch	カメラのデジタル入力信号の積分時間（絶対値）設定値を取得します。
⇒	set_anti_glitch	カメラのデジタル入力信号の積分時間（絶対値）を設定します。
⇒	get_anti_chattering_min_max	カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）の最小値と最大値を取得します。
⇒	get_anti_chattering	カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）設定値を取得します。
⇒	set_anti_chattering	カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）を設定します。
TimerControl		
⇒	get_timer_duration_min_max	カメラの Timer0Active 信号の幅の最小値と最大値を取得します。
⇒	get_timer_duration	カメラの Timer0Active 信号の幅を取得します。
⇒	set_timer_duration	カメラの Timer0Active 信号の幅を設定します。
⇒	get_timer_delay_min_max	カメラの Timer0Active 信号の遅延量の最小値と最大値を取得します。
⇒	get_timer_delay	カメラの Timer0Active 信号の遅延量を取得します。
⇒	set_timer_delay	カメラの Timer0Active 信号の遅延量を設定します。

	名 称	説 明
⇒	get_timer_trigger_source	カメラの Timer0Active 信号の基準信号を取得します。
⇒	set_timer_trigger_source	カメラの Timer0Active 信号の基準信号を設定します。
Gain		
⇒	get_gain_min_max	カメラのゲインの最小値と最大値を取得します。
⇒	get_gain	カメラのゲインを取得します。
⇒	set_gain	カメラのゲインを設定します。
⇒	get_gain_auto	カメラの AGC 動作モードを取得します。
⇒	set_gain_auto	カメラの AGC 動作モードを設定します。
BlackLevel		
⇒	get_black_level_min_max	カメラの映像の黒レベルの最小値と最大値を取得します。
⇒	get_black_level	カメラの映像の黒レベルを取得します。
⇒	set_black_level	カメラの映像の黒レベルを設定します。
Gamma		
⇒	get_gamma_min_max	カメラのガンマ補正値の最小値と最大値を取得します。
⇒	get_gamma	カメラのガンマ補正値を取得します。
⇒	set_gamma	カメラのガンマ補正値を設定します。
WhiteBalance		
⇒	get_balance_ratio_min_max	カメラのホワイトバランスゲイン（倍率）の最小値と最大値を取得します。
⇒	get_balance_ratio	カメラのホワイトバランスゲイン（倍率）を取得します。
⇒	set_balance_ratio	カメラのホワイトバランスゲイン（倍率）を設定します。
⇒	get_balance_white_auto	カメラのホワイトバランスゲイン自動調整モードを取得します。
⇒	set_balance_white_auto	カメラのホワイトバランスゲイン自動調整モードを設定します。
Hue		
⇒	get_hue_min_max	カメラの色相の最小値と最大値を取得します。
⇒	get_hue	カメラの色相の設定値を取得します。
⇒	set_hue	カメラの色相を設定します。
Saturation		
⇒	get_saturation_min_max	カメラの彩度の最小値と最大値を取得します。
⇒	get_saturation	カメラの彩度の設定値を取得します。
⇒	set_saturation	カメラの彩度を設定します。
Sharpness		
⇒	get_sharpness_min_max	カメラの画像のエッジ強度の最小値と最大値を取得します。
⇒	get_sharpness	カメラの画像のエッジ強度の設定値を取得します。
⇒	set_sharpness	カメラの画像のエッジ強度を設定します。
ColorCorrectionMatrix		
⇒	get_color_correction_matrix_min_max	カメラの色補正マトリクスの係数の最小値と最大値を

	名 称	説 明
		取得します。
⇒	get_color_correction_matrix	カメラの色補正マトリクスの係数を取得します。
⇒	set_color_correction_matrix	カメラの色補正マトリクスの係数を設定します。
LUT Control		
⇒	get_lut_enable	カメラの LUT モード（有効 / 無効）を取得します。
⇒	set_lut_enable	カメラの LUT モード（有効 / 無効）を設定します。
⇒	get_lut_value	カメラの LUT の出力値を取得します。
⇒	set_lut_value	カメラの LUT の出力値を設定します。
UserSetControl		
⇒	execute_user_set_load	カメラに実装されている不揮発性メモリ（ユーザーメモリ）から、設定パラメータをカメラにロードします。
⇒	execute_user_set_save	現在カメラに設定されているパラメータを、カメラに実装されている不揮発性メモリ（ユーザーメモリ）にセーブします。
⇒	execute_user_set_quick_save	現在カメラに設定されているパラメータを、カメラに実装されている揮発性メモリ（ユーザーメモリ）にセーブします。
⇒	get_user_set_default	カメラの起動時にロードするユーザー設定チャンネルを読み出します。
⇒	set_user_set_default	カメラの起動時にロードするユーザー設定チャンネルを設定します。
⇒	execute_user_set_save_and_set_default	現在カメラに設定されているパラメータを、カメラに実装されている不揮発性メモリ（ユーザーメモリ）にセーブし、カメラ起動時にユーザー設定チャンネルを設定します。
SequentialShutterControl		
⇒	get_sequential_shutter_enable	カメラのシーケンシャルシャッターモード（有効 / 無効）を取得します。
⇒	set_sequential_shutter_enable	カメラのシーケンシャルシャッターモード（有効 / 無効）を設定します。
⇒	get_sequential_shutter_terminate_at_min_max	カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数の最小値と最大値を取得します。
⇒	get_sequential_shutter_terminate_at	カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数を取得します。
⇒	set_sequential_shutter_terminate_at	カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数を設定します。
⇒	get_sequential_shutter_index_min_max	カメラのシーケンシャルシャッター機能の登録を行うシーケンス番号の最小値と最大値を取得します。
⇒	get_sequential_shutter_entry_min_max	カメラのシーケンシャルシャッター機能のシーケンスに登録するユーザー設定チャンネル（UserSet 番号）の最小値と最大値を取得します。
⇒	get_sequential_shutter_entry	カメラのシーケンシャルシャッター機能のシーケンスに登録されているユーザー設定チャンネル（UserSet 番号）を取得します。
⇒	set_sequential_shutter_entry	カメラのシーケンシャルシャッター機能のシーケンスにユーザー設定チャンネル（UserSet 番号）を登録します。
UserDefinedName		
⇒	get_user_defined_name	カメラのユーザー定義情報を取得します。
⇒	set_user_defined_name	カメラのユーザー定義情報を設定します。

	名 称	説 明
Chunk		
⇒	get_chunk_mode_active	カメラのチャンク機能の有効状態を取得します。
⇒	set_chunk_mode_active	カメラのチャンク機能の有効／無効を設定します。
⇒	get_chunk_enable	カメラのチャンクデータの有効状態を取得します。
⇒	set_chunk_enable	カメラのチャンクデータの有効状態を設定します。
⇒	get_chunk_user_area_length	カメラの ChunkUserAreaTable の長さを取得します。
⇒	get_chunk_user_area_table	カメラの ChunkUserAreaTable に設定されているデータを取得します。
⇒	set_chunk_user_area_table	カメラの ChunkUserAreaTable にデータを設定します。
FrameSynchronization		
⇒	get_frame_synchronization	カメラのフレーム同期制御方法を取得します。
⇒	set_frame_synchronization	カメラのフレーム同期制御方法を設定します。

7.6.1. ImageFormatControl

カメラの映像フォーマット（Format0 ～ Format2）の制御を行います。

映像フォーマットの機能はカメラまたはカメラのファームウェアバージョンにより異なります。

カメラの ImageFormatControl 機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.1.1. get_image_format_selector

カメラの映像フォーマットを取得します。

[構文]

```
pytelicam.CameraControl.get_image_format_selector(self)
```

[戻り値]

tuple 型のデータ（status, format）

status : 実行結果を表すステータスコード

format : 映像フォーマットの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraImageFormat](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ImageFormatSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラによって機能が異なる場合があります。

カメラの ImageFormatControl 機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

使用例を以下に示します。

```
status, fmt = cam_device.cam_control.get_image_format_selector()
print('status={0} , format={1}'.format(status, fmt))
```

7.6.1.2. set_image_format_selector

カメラの映像フォーマットを設定します。

[構文]

```
pytelicam.CameraControl.set_image_format_selector(self, format)
```

[パラメータ]

パラメータ	内 容
format (pytelicam.CameraImageFormat)	映像フォーマットを格納する変数の参照です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ImageFormatSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

カメラによって機能が異なる場合があります。

カメラの ImageFormatControl 機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

使用例を以下に示します。

```
status = cam_device.cam_control.set_image_format_selector \
(pytelicam.pytelicam.CameraImageFormat.Format0)
print('status={0}'.format(status))
```

7.6.2. Scalable

カメラのスケラブル機能の設定を行います。

スケラブル読み出しは、最大映像出力有効画素領域のうち任意の矩形領域のみを読み出し、出力する方法です。 垂直方向(縦方向)の不要な領域を高速で読み飛ばすことでフレームレートを向上させることができます。

選択できる形状は連続したユニット単位の矩形形状のみで、凸や凹のような選択はできません。また選択できるウィンド数は1個です。

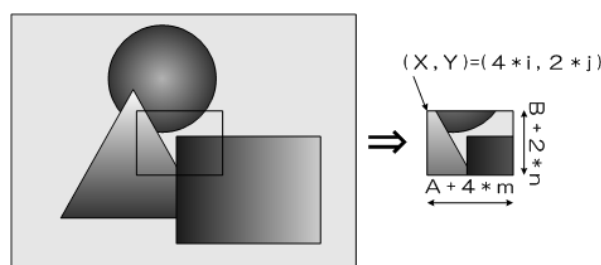
・ウィンドのサイズ : $\{A + 4 \times m (H)\} \times \{B + 2 \times n (V)\}$

※ A, B はそれぞれの最小ユニットサイズ

※ m, n は整数、但しウィンドが最大ユニットサイズの全画面からはみ出さないこと

・ウィンドの開始位置 : $\{4 \times i (H)\} \times \{2 \times j (V)\}$

※ i, j は整数、但しウィンドが最大ユニットサイズの全画面からはみ出さないこと



カメラのスケラブル機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.2.1. get_sensor_width

カメラの水平有効画素数を取得します。

[構文]

```
pytelicam.CameraControl.get_sensor_width(self)
```

[戻り値]

tuple 型のデータ (status, sensor_width)

status : 実行結果を表すステータスコード

sensor_width : 水平有効画素数

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.2. get_sensor_height

カメラの垂直有効画素数を取得します。

[構文]

pytelicam.CameraControl.get_sensor_height (self)

[戻り値]

tuple 型のデータ (status, sensor_height)

status : 実行結果を表すステータスコード

sensor_height : 垂直有効画素数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.3. get_roi

カメラの ROI（領域）を取得します。

[構文]

pytelicam.CameraControl.get_roi (self)

[戻り値]

tuple 型のデータ (status, width, height, offset_x, offset_y)

status : 実行結果を表すステータスコード

width : 映像の幅

height : 映像の高さ

offset_x : 映像の水平方向開始位置

offset_y : 映像の垂直方向開始位置

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int, int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.4. set_roi

カメラの ROI（領域）を設定します。

[構文]

```
pytelicam.CameraControl.set_roi(self, width, height, offset_x, offset_y)
```

[パラメータ]

パラメータ	内 容
width (int)	映像の幅です。
height (int)	映像の高さです。
offset_x (int)	映像の水平方向開始位置です。
offset_y (int)	映像の垂直方向開始位置です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本映像ストリーム出力中は設定を変更することができません。

ただし、映像ストリーム出力中でも offsetx, offsety のみ変更することができるカメラがあります。
カメラのスケラブル機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.2.5. get_width_min_max

カメラに設定できる映像の幅の最小値・最大値・増加量を取得します。

[構文]

```
pytelicam.CameraControl.get_width_min_max (self)
```

[戻り値]

tuple 型のデータ (status, width_min, width_max, width_inc)

status : 実行結果を表すステータスコード

width_min : 最小値

width_max : 最大値

width_inc : 増加量

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.6. get_width

カメラの映像の幅を取得します。

[構文]

```
pytelicam.CameraControl.get_width (self)
```

[戻り値]

tuple 型のデータ (status, width)

status : 実行結果を表すステータスコード

width : 映像の幅

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.7. set_width

カメラの映像の幅を設定します。

[構文]

```
pytelicam.CameraControl.set_width(self, width)
```

[パラメータ]

パラメータ	内 容
width (int)	映像の幅です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

映像ストリーム出力中は設定を変更することはできません。

7.6.2.8. get_height_min_max

カメラに設定できる映像の高さの最小値・最大値・増加量を取得します。

[構文]

```
pytelicam.CameraControl.get_height_min_max (self)
```

[戻り値]

tuple 型のデータ (status, height_min, height_max, height_inc)

status : 実行結果を表すステータスコード

height_min : 最小値

height_max : 最大値

height_inc : 増加量

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int, int, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.9. get_height

カメラの映像の高さを取得します。

[構文]

```
pytelicam.CameraControl.get_height (self)
```

[戻り値]

tuple 型のデータ (status, height)

status : 実行結果を表すステータスコード

height : 映像の高さ

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.10. set_height

カメラの映像の高さを設定します。

[構文]

```
pytelicam.CameraControl.set_height(self, height)
```

[パラメータ]

パラメータ	内 容
height (int)	映像の高さです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

映像ストリーム出力中は設定を変更することはできません。

7.6.2.11. get_offset_x_min_max

カメラに設定できる映像の水平方向開始位置の最小値・最大値・増加量を取得します。

[構文]

```
pytelicam.CameraControl.get_offset_x_min_max (self)
```

[戻り値]

tuple 型のデータ (status, offset_x_min, offset_x_max, offset_x_inc)

status : 実行結果を表すステータスコード
offset_x_min : 最小値
offset_x_max : 最大値
offset_x_inc : 増加量

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int, int, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.12. get_offset_x

カメラの映像の水平方向開始位置を取得します。

[構文]

```
pytelicam.CameraControl.get_offset_x (self)
```

[戻り値]

tuple 型のデータ (status, offset_x)

status : 実行結果を表すステータスコード
offset_x : 映像の水平方向開始位置

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.13. set_offset_x

カメラの映像の水平方向開始位置を設定します。

[構文]

```
pytelicam.CameraControl.set_offset_x(self, offset_x)
```

[パラメータ]

パラメータ	内 容
offset_x (int)	映像の水平方向開始位置です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

映像ストリーム出力中に設定できるかどうかはカメラにより異なります。

カメラのスケラブル機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.2.14. get_offset_y_min_max

カメラに設定できる映像の垂直方向開始位置の最小値・最大値・増加量を取得します。

[構文]

```
pytelicam.CameraControl.get_offset_y_min_max (self)
```

[戻り値]

tuple 型のデータ (status, offset_y_min, offset_y_max, offset_y_inc)

status : 実行結果を表すステータスコード

offset_y_min : 最小値

offset_y_max : 最大値

offset_y_inc : 増加量

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.15. get_offset_y

カメラの映像の垂直方向開始位置を取得します。

[構文]

```
pytelicam.CameraControl.get_offset_y (self)
```

[戻り値]

tuple 型のデータ (status, offset_y)

status : 実行結果を表すステータスコード

offset_y : 映像の水平方向開始位置

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.2.16. set_offset_y

カメラの映像の垂直方向開始位置を設定します。

[構文]

```
pytelicam.CameraControl.set_offset_y(self, offset_y)
```

[パラメータ]

パラメータ	内 容
offset_y (int)	映像の垂直方向開始位置です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

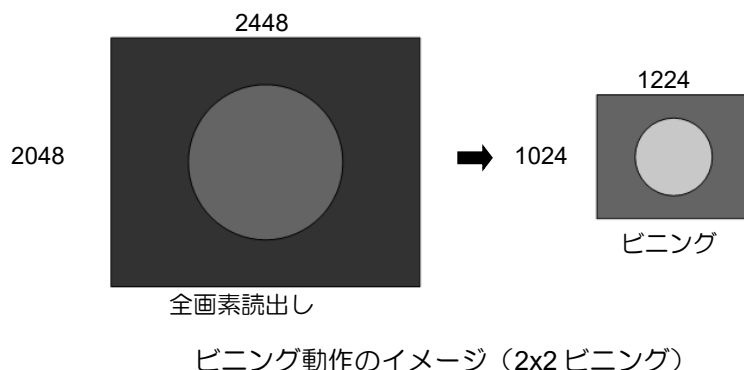
映像ストリーム出力中に設定できるかどうかはカメラにより異なります。

カメラのスケラブル機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3. Binning

カメラのビニング機能の制御を行います。

ビニング読み出しでは隣接する画素を加算することで、画素単位の感度が向上します。
さらにインターフェース帯域幅の占有帯域の軽減とフレームレートを向上させることができます。



カメラのビニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.1. get_binning_horaizontal_min_max

カメラに設定できる水平方向ビニング設定値の最小値・最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_binning_horizaontal_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

BinningHorizontal レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

ビニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.2. get_binning_horizontal

カメラの水平方向ビニング設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_binning_horizontal (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 水平方向ビニング設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

BinningHorizontal レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

ビニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.3. set_binning_horizontal

カメラの水平方向ビニング設定値を設定します。

[構文]

```
pytelicam.CameraControl.set_binning_horizontal(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	水平方向ビニング設定値です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

BinningHorizontal レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

ビニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.4. get_binning_vertical_min_max

カメラに設定できる垂直方向ビンニング設定値の最小値・最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_binning_vertical_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値

max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

BinningVertical レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

ビンニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビンニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.5. get_binning_vertical

カメラの垂直方向ビンニング設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_binning_vertical (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 垂直方向ビンニング設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

BinningVertical レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

ビンニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビンニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.3.6. set_binning_vertical

カメラの垂直方向ビニング設定値を設定します。

【構文】

```
pytelicam.CameraControl.set_binning_vertical(self, value)
```

【パラメータ】

パラメータ	内 容
value (int)	垂直方向ビニング設定値です。

【戻り値】

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

【戻り値の型】

[pytelicam.CamApiStatus](#)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

【備考】

BinningVertical レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

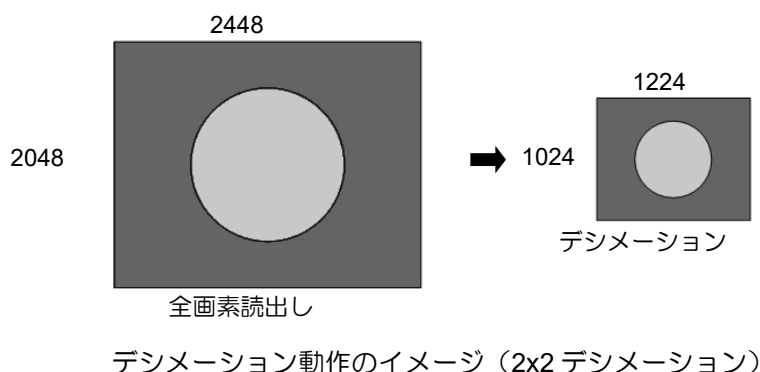
ビニング機能のパラメータを取得・設定できる条件はカメラにより異なります。

カメラのビニング機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.4. Decimation

カメラのデシメーション機能の制御を行います。

デシメーション機能は読み出しラインを間引くことにより全有効エリアを高速で読み出し、インターフェース帯域幅の占有帯域の軽減とフレームレートを向上させることができます。



カメラのデシメーション機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.4.1. get_decimation_horizontal_min_max

カメラに設定できる水平方向デシメーション設定値の最小値・最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_decimation_hori_aontal_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.4.2. get_decimation_horizontal

カメラの水平方向デシメーション設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_decimation_horizontal (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 水平方向デシメーション設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.4.3. set_decimation_horizontal

カメラの水平方向デシメーション設定値を設定します。

[構文]

```
pytelicam.CameraControl.set_decimation_horizontal(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	水平方向デシメーション設定値です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

7.6.4.4. get_decimation_vertical_min_max

カメラに設定できる垂直方向デシメーション設定値の最小値・最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_decimation_vertical_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status	: 実行結果を表すステータスコード
min	: 最小値
max	: 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

DecimationHorizontal レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

デシメーション機能のパラメータを取得・設定できる条件はカメラにより異なります。
カメラのデシメーション機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.4.5. get_decimation_vertical

カメラの垂直方向デシメーション設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_decimation_vertical (self)
```

[戻り値]

tuple 型のデータ (status, value)

status	: 実行結果を表すステータスコード
value	: 垂直方向ビニング設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

DecimationVertical レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

デシメーション機能のパラメータを取得・設定できる条件はカメラにより異なります。
カメラのデシメーション機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.4.6. set_decimation_vertical

カメラの垂直方向デシメーション設定値を設定します。

[構文]

```
pytelicam.CameraControl.set_decimation_vertical(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	垂直方向デシメーション設定値です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

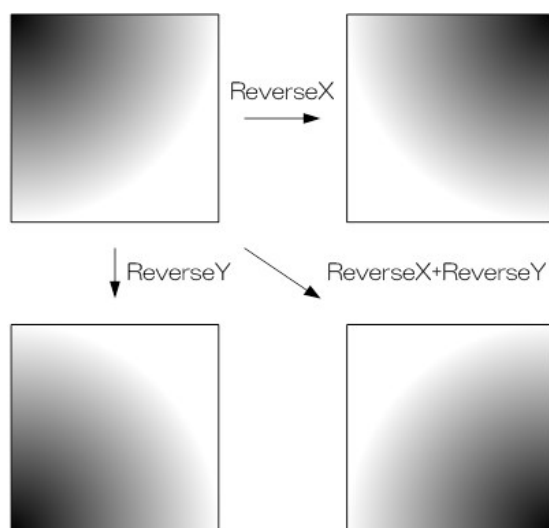
DecimationVertical レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

デシメーション機能のパラメータを取得・設定できる条件はカメラにより異なります。
カメラのデシメーション機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.5. Reverse

カメラの映像反転機能の制御を行います。



カメラの映像反転機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.5.1. get_reverse_x

カメラの水平方向の映像反転設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_reverse_x (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 水平方向の映像反転設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ReverseX レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.5.2. set_reverse_x

カメラの水平方向の映像反転 ON/OFF を設定します。

[構文]

```
pytelicam.CameraControl.set_reverse_x(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	水平方向の映像反転設定値を格納する変数の参照です。 true の場合 反転 ON、false の場合 反転 OFF です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ReverseX レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.5.3. get_reverse_y

カメラの垂直方向の映像反転設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_reverse_y (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 水平方向の映像反転設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ReverseY レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.5.4. set_reverse_y

カメラの垂直方向の映像反転 ON/OFF を設定します。

[構文]

```
pytelicam.CameraControl.set_reverse_y(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	垂直方向の映像反転設定値を格納する変数の参照です。 true の場合 反転 ON、false の場合 反転 OFF です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ReverseY レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.6. PixelFormat

カメラの映像ストリームのピクセルフォーマットの制御を行います。
カメラの映像ストリームのピクセルフォーマットに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.6.1. get_pixel_format

カメラの映像ストリームのピクセルフォーマットを取得します。

[構文]

```
pytelicam.CameraControl.get_pixel_format(self)
```

[戻り値]

tuple 型のデータ (status, pixel_format)

status : 実行結果を表すステータスコード

pixel_format : ピクセルフォーマットの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraPixelFormat](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

PixelFormat、または PixelCoding と PixelSize レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.6.2. set_pixel_format

カメラの映像ストリームのピクセルフォーマットを設定します。

[構文]

```
pytelicam.CameraControl.set_pixel_format(self, pixel_format)
```

[パラメータ]

パラメータ	内 容
pixel_format (pytelicam.CameraPixelFormat)	ピクセルフォーマットです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

PixelFormat、または PixelCoding と PixelSize レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.7. TestPattern

カメラのテストパターン機能の制御を行います。

カメラのテストパターン機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.7.1. get_test_pattern

カメラのテストパターン設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_test_pattern(self)
```

[戻り値]

tuple 型のデータ (status, test_pattern)

status : 実行結果を表すステータスコード

test_pattern : ピクセルフォーマットの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraTestPattern](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TestPattern または TestImageSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.7.2. set_test_pattern

カメラのテストパターンを設定します。

[構文]

```
pytelicam.CameraControl.set_test_pattern(self, test_pattern)
```

[パラメータ]

パラメータ	内 容
test_pattern (pytelicam.CameraTestPattern)	ピクセルフォーマットです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TestPattern または TestImageSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8. AcquisitionControl

カメラの映像出力についての実行・設定を行います。

カメラの AcquisitionControl 機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.8.1. get_stream_payload_size

カメラのレジスタから、映像ストリームペイロードサイズ（画像サイズ）を取得します。

[構文]

```
pytelicam.CameraControl.get_stream_payload_size (self)
```

[戻り値]

tuple 型のデータ (status, payload_size)

status : 実行結果を表すステータスコード

payload_size : 映像ストリームペイロードサイズ（画像サイズ）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.6.8.2. get_stream_enable

カメラのレジスタから、ストリーム状態を取得します。

[構文]

```
pytelicam.CameraControl.get_stream_enable (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ストリームの状態

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本メソッドは USB3 カメラでのみ使用することができます。

7.6.8.3. get_acquisition_frame_count_min_max

"AcquisitionFrameCount"の最小値と最大値を取得します。

"AcquisitionFrameCount"は、マルチフレーム映像ストリーム転送モードおよびカメライメージバッファ転送モード時の映像ストリーム転送枚数です。

[構文]

```
pytelicam.CameraControl.get_acquisition_frame_count_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値

max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8.4. get_acquisition_frame_count

"AcquisitionFrameCount" の値を取得します。

マルチフレーム映像ストリーム転送モード時は、転送したフレームの枚数が "AcquisitionFrameCount" の値に達すると、カメラは画像取得を停止します。

カメライメージバッファ転送モード時は、ExecuteImageBufferRead() が実行されるたびに、カメラはイメージバッファから "AcquisitionFrameCount" で設定された枚数のフレームを転送します。

[構文]

```
pytelicam.CameraControl.get_acquisition_frame_count (self)
```

[戻り値]

tuple 型のデータ (status, frame_count)

status : 実行結果を表すステータスコード

frame_count : 映像ストリーム転送枚数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8.5. set_acquisition_frame_count

“AcquisitionFrameCount” の値を設定します。

マルチフレーム映像ストリーム転送モード時は、転送したフレームの枚数が “AcquisitionFrameCount” の値に達すると、カメラは画像取得を停止します。

カメライメージバッファ転送モード時は、ExecuteImageBufferRead() が実行されるたびに、カメラはイメージバッファから “AcquisitionFrameCount” で設定された枚数のフレームを転送します。

【構文】

```
pytelicam.CameraControl.set_acquisition_frame_count(self, frame_count)
```

【パラメータ】

パラメータ	内 容
frame_count (int)	映像ストリーム転送枚数です。

【戻り値】

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

【戻り値の型】

[pytelicam.CamApiStatus](#)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

【備考】

AcquisitionFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.8.6. get_acquisition_frame_rate_control

映像のフレームレート設定を取得します。

[構文]

```
pytelicam.CameraControl.get_acquisition_frame_rate_control(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : フレームレート設定の種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraAcqFrameRateCtrl](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameRate レジスタ (または AcquisitionFrameRateControl /

AcquisitionFrameRateEnable ノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8.7. set_acquisition_frame_rate_control

映像のフレームレート設定を設定します。

[構文]

```
pytelicam.CameraControl.set_acquisition_frame_rate_control(self, value)
```

[パラメータ]

パラメータ	内 容
value (pytelicam.CameraAcqFrameRateCtrl)	フレームレート設定です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameRate レジスタ（または AcquisitionFrameRateControl / AcquisitionFrameRateEnable ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

[CameraAcqFrameRateCtrl.NoSpecify](#) を設定した場合、AcquisitionFrameRate レジスタ（またはノード）の値が変わる場合があります。使用するカメラの動作を確認してからご使用ください。

7.6.8.8. get_acquisition_frame_rate_min_max

映像のフレームレート最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_acquisition_frame_rate_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameRate レジスタ (または AcquisitionFrameRateControl / AcquisitionFrameRateEnable ノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8.9. get_acquisition_frame_rate

映像のフレームレート設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_acquisition_frame_rate (self)
```

[戻り値]

tuple 型のデータ (status, frame_rate)

status : 実行結果を表すステータスコード
frame_rate : 映像ストリーム転送枚数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameRate レジスタ (または AcquisitionFrameRateControl / AcquisitionFrameRateEnable ノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.8.10. set_acquisition_frame_rate

映像のフレームレートを設定します。

[構文]

```
pytelicam.CameraControl.set_acquisition_frame_rate(self, frame_rate)
```

[パラメータ]

パラメータ	内 容
frame_rate (double)	映像ストリーム転送枚数です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.8.11. execute_acquisition_start

AcquisitionStart コマンドを実行します。

シングルフレーム映像ストリーム転送モードおよびマルチフレーム映像ストリーム転送モード時に再度画像を取得するときに実行します。

[構文]

```
pytelicam.CameraControl.execute_acquisition_start(self)
```

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionCommand レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

シングルフレーム映像ストリーム転送モードまたはマルチフレーム映像ストリーム転送モードでストリーミングを実行している時以外は本メソッドを実行しないでください。

7.6.8.12. get_high_framerate_mode

カメラの高フレームレートモード（HighFramerateMode）を取得します。

一部のカラーモデルのカメラは高フレームレートモード（HighFramerateMode）を有しています。高フレームレートモードを利用することにより、フレームレートを向上させることができます。

【構文】

```
pytelicam.CameraControl.get_high_framerate_mode (self)
```

【戻り値】

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : HighFramerateMode

【戻り値の型】

tuple([pytelicam.CamApiStatus](#), bool)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

【備考】

HighFramerateMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラの高フレームレートモードに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.8.13. set_high_framerate_mode

カメラの高フレームレートモード（HighFramerateMode）を設定します。
一部のカラーモデルのカメラは高フレームレートモード（HighFramerateMode）を有しています。 高フレームレートモードを利用することにより、フレームレートを向上させることができます。

[構文]

```
pytelicam.CameraControl.set_high_framerate_mode(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	設定する HighFramerateMode です。 false の場合 OFF、true の場合 ON です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

HighFramerateMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

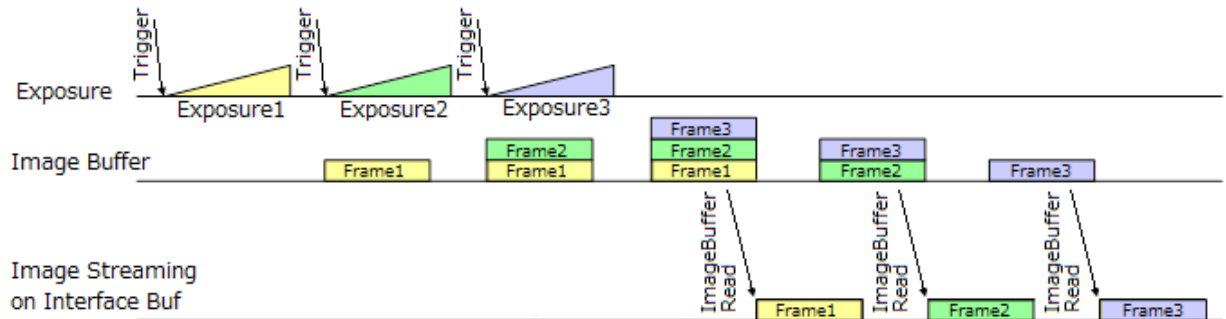
カメラの高フレームレートモードに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.9. ImageBuffer

カメラのイメージバッファ機能の制御を行います。

ImageBuffer はイメージバッファに画像を取り込んでおき、任意のタイミングで読み出しを行うことができます。

この機能はノーマルシャッターモードでも動作しますが、通常ランダムトリガモードにて使用します。



カメラのイメージバッファ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.9.1. get_image_buffer_mode

カメラのイメージバッファモード設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_image_buffer_mode (self)
```

[戻り値]

tuple 型のデータ (status, mode)

status : 実行結果を表すステータスコード
mode : イメージバッファモード設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraImageBufferMode](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ImageBufferMode レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.9.2. set_image_buffer_mode

カメラのイメージバッファモード ON/OFF を設定します。

[構文]

```
pytelicam.CameraControl.set_image_buffer_mode(self, mode)
```

[パラメータ]

パラメータ	内 容
mode (pytelicam.CameraImageBufferMode)	イメージバッファモード ON/OFF です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ImageBufferMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中は設定を変更することはできません。

7.6.9.3. get_image_buffer_frame_count

カメラのイメージバッファに取り込まれたフレーム数を取得します。

[構文]

```
pytelicam.CameraControl.get_image_buffer_frame_count (self)
```

[戻り値]

tuple 型のデータ (status, frame_count)

status : 実行結果を表すステータスコード

frame_count : 映像ストリーム転送枚数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ImageBufferFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.9.4. execute_image_buffer_read

カメラのイメージバッファの画像を読み出します。

[構文]

```
pytelicam.CameraControl.execute_image_buffer_read(self)
```

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AcquisitionCommand レジスタ（または ImageBufferRead ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラの AcquisitionMode レジスタに "ImageBufferRead" を設定します。

1 回のコマンド実行でカメラが出力するフレーム数は、[SetAcquisitionFrameCount\(\)](#) で設定できます。

カメラから出力されたフレームは、[CameraStream クラス](#) を使用して取り込んでください。

カメラのイメージバッファ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10. TriggerControl

カメラのトリガ機能の制御を行います。

カメラの露光動作には、フリーランで動作するノーマルシャッターモードと外部からのトリガにより任意のタイミングで動作するランダムトリガシャッターモードの2種類があります。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.1. get_trigger_mode

カメラのトリガ動作モード（TriggerMode）を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_mode (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : トリガ動作モード

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

トリガ動作モードが OFF の場合、ノーマルシャッターモードで動作します。フリーラン（内部同期）モードが同期モードとして使用され、露光時間は ExposureTime レジスタで制御されます。

トリガ動作モードが ON の場合、ランダムシャッターモードで動作します。ハードウェアトリガモードまたはソフトウェアトリガモードが同期モードとして使用され、露光時間は TriggerSequence 等の他のレジスタによって設定される様々なモードで制御されます。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.2. set_trigger_mode

カメラのトリガ動作モード（TriggerMode）を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_mode(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	トリガ動作モードです。 false の場合 トリガ動作モード OFF、true の場合 トリガ動作モード ON です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

映像ストリーム出力中に設定できるかどうかはカメラにより異なります。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.3. get_trigger_sequence

カメラのランダムトリガシャッタの露光時間制御モード（TriggerSequence）を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_sequence(self)
```

[戻り値]

tuple 型のデータ (status, sequence)

status : 実行結果を表すステータスコード

sequence : 露光時間制御モードの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraTriggerSequence](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerSequence、または ExposureMode と TriggerSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラによって、読み出すレジスタが異なります。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.4. set_trigger_sequence

カメラのランダムトリガシャッタの露光時間制御モード（TriggerSequence）を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_sequence(self, sequence)
```

[パラメータ]

パラメータ	内 容
sequence (pytelicam.CameraTriggerSequence)	露光時間制御モードです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerSequence、または ExposureMode と TriggerSelector レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラによって、設定するレジスタが異なります。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.5. get_trigger_source

カメラのランダムトリガシャッタのトリガソース（TriggerSource）を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_source(self)
```

[戻り値]

tuple 型のデータ（status, source）

status : 実行結果を表すステータスコード

source : トリガソースの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraTriggerSource](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.10.6. set_trigger_source

カメラのランダムトリガシャッタのトリガソース（TriggerSource）を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_source(self, source)
```

[パラメータ]

パラメータ	内 容
source (pytelicam.CameraTriggerSource)	設定するトリガソースです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.10.7. `get_trigger_additional_parameter_min_max`

Bulk (FrameBurst) モード動作設定時の露光回数の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_additional_parameter_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

Bulk (FrameBurst) モード動作 設定時、カメラは 1 回のトリガで設定された枚数のフレームを転送します。

本メソッドは、TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタに設定できる最小値と最大値を取得します。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.8. `get_trigger_additional_parameter`

Bulk (FrameBurst) モード動作設定時の露光回数を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_additional_parameter (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 露光回数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

Bulk (FrameBurst) モード動作 設定時、カメラは 1 回のトリガで、設定された枚数のフレームを転送します。

本メソッドは、TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタに設定されている値を取得します。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

TeliCamAPI.h をインクルードする必要があります。

7.6.10.9. set_trigger_additional_parameter

Bulk (FrameBurst) モード動作設定時の露光回数を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_additional_parameter (self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	設定する露光回数です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

Bulk (FrameBurst) モード動作 設定時、カメラは 1 回のトリガで、設定された枚数のフレームを転送します。

本メソッドは、TriggerAdditionalParameter または AcquisitionBurstFrameCount レジスタに値を設定します。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

TeliCamAPI.h をインクルードする必要があります。

7.6.10.10. get_trigger_delay_min_max

カメラのトリガ信号検出から露光開始までの遅延量の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_delay_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: マイクロ秒)

max : 最大値 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerDelay レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.10.11. get_trigger_delay

カメラのトリガ信号検出から露光開始までの遅延量を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_delay (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : 遅延量 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerDelay レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.10.12. set_trigger_delay

カメラのトリガ信号検出から露光開始までの遅延量を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_delay(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	設定する遅延量です。(単位：マイクロ秒)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerDelay レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.10.13. execute_software_trigger

カメラのソフトウェアトリガを実行します。

[構文]

```
pytelicam.CameraControl.execute_software_trigger(self)
```

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

SoftwareTrigger レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

正常終了しても、カメラがソフトウェアトリガ受付可能状態でない場合は何も実行されません。

7.6.10.14. get_trigger_activation

カメラのハードウェアトリガの有効エッジ（TriggerActivation）を取得します。

[構文]

```
pytelicam.CameraControl.get_trigger_activation(self)
```

[戻り値]

tuple 型のデータ（status, activation）

status : 実行結果を表すステータスコード

activation : ハードウェアトリガの有効エッジ

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraTriggerActivation](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TriggerActivation レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

取得される値は、現在設定されているランダムトリガシャッタのトリガソース設定（TriggerSource）に対する有効エッジです。本メソッドを実行する前に、[set_trigger_source\(\)](#)によりランダムトリガシャッタのトリガソースを設定してください。

LineInverterAll レジスタが存在するカメラでは、本メソッドを実行すると LineInverterAll レジスタの値を読み出し、対象のラインの値を取得します。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.10.15. set_trigger_activation

カメラのハードウェアトリガの有効エッジ（TriggerActivation）を設定します。

[構文]

```
pytelicam.CameraControl.set_trigger_activation(self, activation)
```

[パラメータ]

パラメータ	内 容
activation (pytelicam.CameraTriggerActivation)	設定するハードウェアトリガの有効エッジです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TriggerActivation または LineInverterAll レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

設定する値は、現在設定されているランダムトリガシャッタのトリガソース設定（TriggerSource）に対する有効エッジです。本メソッドを実行する前に、[set_trigger_source\(\)](#) によりランダムトリガシャッタのトリガソースを設定してください。

LineInverterAll レジスタが存在するカメラでは、LineInverterAll レジスタの値を読み出し、対象のラインの値のみを変更して LineInverterAll レジスタに値を設定します。[set_line_inverter_all\(\)](#) または [set_line_inverter\(\)](#) を実行すると、本メソッドで設定した有効エッジが変更される場合があります。

カメラのトリガ機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.11. ExposureTime

カメラの露光時間の制御を行います。

カメラの露光時間制御機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.11.1. get_exposure_time_control

カメラの露光時間制御モードの設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_exposure_time_control(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 制御モード設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraExposureTimeCtrl](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ExposureTimeControl または ExposureAuto レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラによって機能およびアクセスするレジスタが異なる場合があります。

カメラの露光時間制御機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.11.2. set_exposure_time_control

カメラの露光時間制御モードを設定します。

[構文]

```
pytelicam.CameraControl.set_exposure_time_control(self, value)
```

[パラメータ]

パラメータ	内 容
value (pytelicam.CameraExposureTimeCtrl)	設定する制御モードです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ExposureTimeControl または ExposureAuto レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラによって機能およびアクセスするレジスタが異なる場合があります。

カメラの露光時間制御機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.11.3. get_exposure_time_min_max

カメラの露光時間制御モードが Manual に設定されているときの、露光時間の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_exposure_time_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: マイクロ秒)

max : 最大値 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ExposureTime レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.11.4. get_exposure_time

カメラの露光時間制御モードが Manual に設定されているときの、露光時間を取得します。

[構文]

```
pytelicam.CameraControl.get_exposure_time (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : 露光時間 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ExposureTime レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.11.5. set_exposure_time

カメラの露光時間制御モードが Manual に設定されているときの、露光時間を設定します。

[構文]

```
pytelicam.CameraControl.set_exposure_time(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	設定する露光時間です。（単位：マイクロ秒）

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ExposureTime レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.11.6. `get_short_exposure_mode`

カメラの超短時間露光モード（ShortExposureMode）を取得します。

一部のカラーモデルのカメラは超短時間露光モード（ShortExposureMode）を有しています。超短時間露光モードを ON に設定すると、マニュアル露光時間制御（Manual）時に高速露光時間設定が可能となります。

[構文]

```
pytelicam.CameraControl.get_short_exposure_mode (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ShortExposureMode

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ShortExposureMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラの超短時間露光モードに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.11.7. set_short_exposure_mode

カメラの超短時間露光モード（ShortExposureMode）を設定します。

一部のカラーモデルのカメラは超短時間露光モード（ShortExposureMode）を有しています。超短時間露光モードを ON に設定すると、マニュアル露光時間制御（Manual）時に高速露光時間設定が可能となります。

【構文】

```
pytelicam.CameraControl.set_short_exposure_mode(self, value)
```

【パラメータ】

パラメータ	内 容
value (bool)	ShortExposureMode を格納する変数の参照です。 false の場合 OFF、true の場合 ON です。

【戻り値】

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

【戻り値の型】

[pytelicam.CamApiStatus](#)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

【備考】

ShortExposureMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラの超短時間露光モードに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12. DigitalloControl

カメラのデジタル I/O の制御を行います。

カメラのデジタル I/O に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.1. get_line_mode_all

カメラの全ラインに対する入出力設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_line_mode_all(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 入出力設定値

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, int)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineModeAll、または LineSelector と LineMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineModeAll レジスタが存在しないカメラでは、各ラインの入出力設定値を読み出し、合成して出力します。

カメラのデジタル I/O に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.2. set_line_mode_all

カメラの全ラインに対する入出力を設定します。

[構文]

```
pytelicam.CameraControl.set_line_mode_all(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	取得した入出力設定値を格納する変数の参照です。 各 bit が各ラインに対応しています。 (Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , ...) bit データが 0 のラインは入力、1 のラインは出力です。 (puiValue = 0x06 のとき、Line0 : 入力 , Line1 : 出力 , Line2 : 出力 , ...)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LineModeAll 、または LineSelector と LineMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineModeAll レジスタが存在しないカメラでは、各ラインの入出力設定値を読み出し、合成して出力します。

カメラのデジタル I/O に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.3. get_line_mode

カメラのラインの入出力を取得します。

[構文]

```
pytelicam.CameraControl.get_line_mode (self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。

[戻り値]

tuple 型のデータ (status, mode)

status : 実行結果を表すステータスコード

mode : 入出力設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraLineMode](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineModeAll、または LineSelector と LineMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineModeAll レジスタが存在するカメラでは、LineModeAll レジスタの値を読み出し、対象のラインの値を取得します。

デジタル I/O 制御機能の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.4. set_line_mode

カメラのラインの入出力を設定します。

[構文]

```
pytelicam.CameraControl.set_line_mode(self, selector, mode)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。
mode (pytelicam.CameraLineMode)	設定する入出力です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LineModeAll、または LineSelector と LineMode レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineModeAll レジスタが存在するカメラでは、LineModeAll レジスタの値を読み出し、対象のラインの値のみを変更して LineModeAll レジスタに値を設定します。

カメラによって機能および書き込み可能な Line が異なる場合があります。

デジタル I/O 制御機能の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.5. get_line_inverter_all

カメラの全ラインに対する極性設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_line_inverter_all(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 極性設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineInverterAll、または LineSelector と LineInverter レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineInverterAll レジスタが存在しないカメラでは、各ラインの極性設定値を読み出し、合成して出力します。

カメラのデジタル I/O に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.6. set_line_inverter_all

カメラの全ラインに対する極性を設定します。

[構文]

```
pytelicam.CameraControl.set_line_inverter_all(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	設定する極性の値です。 各 bit が各ラインに対応しています。(Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , ...) bit データが 0 のラインは Invert なし、1 のラインは Invert あり です。 (puiValue = 0x02 のとき、Line0: Invert なし , Line1 : Invert あり , Line2 : Invert なし , ...)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LineInverterAll、または LineSelector と LineInverter レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineInverterAll レジスタが存在しないカメラでは、各ライン毎に極性設定を行います。

カメラによって機能および書き込み可能な Line が異なる場合があります。

書き込みができない Line の bit データに 1（Invert あり）が設定されている場合、エラーステータス（InvalidParameter）がリターンされる場合があります。

デジタル I/O 制御機能の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.7. get_line_inverter

カメラのラインの極性を取得します。

[構文]

```
pytelicam.CameraControl.get_line_inverter(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。

[戻り値]

tuple 型のデータ (status, inverter)

status : 実行結果を表すステータスコード

invert : 極性設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineInverterAll、または LineSelector と LineInverter レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineInverterAll レジスタが存在するカメラでは、LineInverterAll レジスタの値を読み出し、対象のラインの値を取得します。

デジタル I/O 制御機能の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.8. set_line_inverter

カメラのラインの極性を設定します。

[構文]

```
pytelicam.CameraControl.set_line_inverter(self, selector, invert)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。
invert (bool)	設定する極性の値です。 false の場合 Invert なし、true の場合 invert あり です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LineInverterAll、または LineSelector と LineInverter レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LineInverterAll レジスタが存在するカメラでは、LineInverterAll レジスタの値を読み出し、対象のラインの値のみを変更して LineInverterAll レジスタに値を設定します。

カメラによって機能および書き込み可能な Line が異なる場合があります。

デジタル I/O 制御機能の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.12.9. get_line_status_all

カメラの全ラインに対する現在の状態を取得します。

[構文]

```
pytelicam.CameraControl.get_line_status_all(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の状態を表す値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineInverterAll、または LineSelector と LineInverter レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.10. get_line_status

カメラのラインの現在の状態を取得します。

[構文]

```
pytelicam.CameraControl.get_line_status(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の状態を表す値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineStatusAll、または LineSelector と LineStatus レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.11. get_user_output_value_all

カメラの全ラインに対するユーザー出力設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_user_output_value_all(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ユーザー出力設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

UserOutputValueAll レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.12. set_user_output_value_all

カメラの全ラインに対するユーザー出力設定値を設定します。

[構文]

```
pytelicam.CameraControl.set_user_output_value_all(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	設定するユーザー出力設定値です。 各 bit が各ラインに対応しています。(Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , ...) bit データが 0 のラインは Low、1 のラインは High です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserOutputValueAll レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

ラインの信号種類（LineSource）が UserOutput に設定されているライン以外の bit（出力設定値）は無視されます。

7.6.12.13. get_user_output_value

カメラのラインのユーザー出力設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_line_user_output_value(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の状態を表す値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

UserOutputValueAll レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.14. set_user_output_value

カメラのラインのユーザー出力設定値を設定します。

[構文]

```
pytelicam.CameraControl.set_user_output_value(self, selector, value)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。
value (bool)	設定するユーザー出力設定値です。 false の場合 Low、true の場合 High です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserOutputValueAll レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.15. get_line_source

カメラのラインの信号種類を取得します。

[構文]

```
pytelicam.CameraControl.get_line_source(self, source)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。

[戻り値]

tuple 型のデータ (status, source)

status : 実行結果を表すステータスコード

source : 信号種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), pytelicam.CameraLineSource)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LineSelector と LineSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.12.16. set_line_source

カメラのラインの信号種類を設定します。

[構文]

```
pytelicam.CameraControl.set_line_source(self, selector, source)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraLineSelector)	設定するラインです。
source (pytelicam.CameraLineSource)	設定する信号種類です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LineSelector と LineSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラにより、設定できる信号種類が異なります。

カメラのデジタル I/O に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

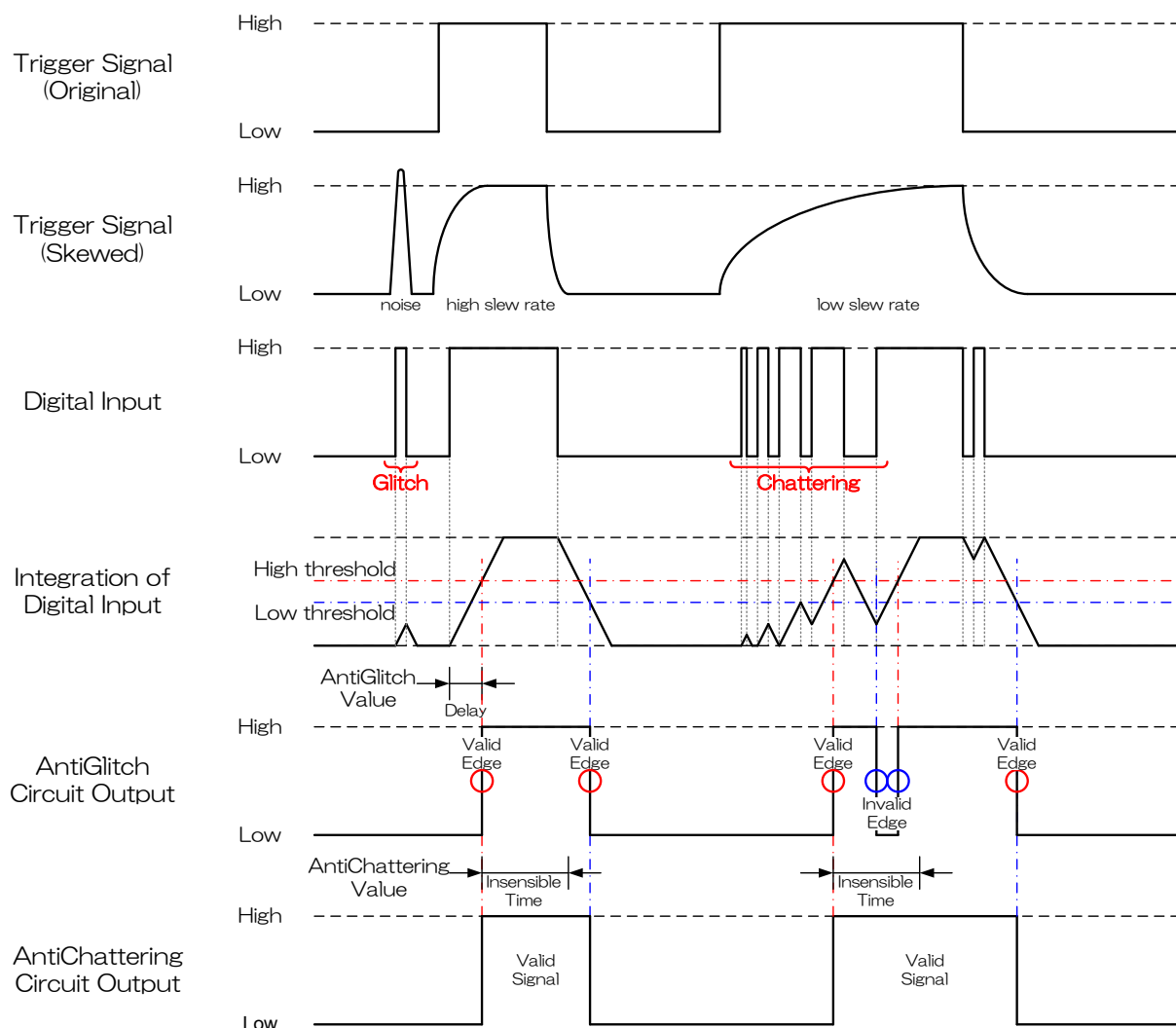
7.6.13. AntiClitch / AntiChattering

カメラのアンチグリッチとアンチチャタリングの制御を行います。

アンチグリッチとアンチチャタリングはノイズや不安定なデジタル入力（トリガ信号）にフィルタをかける機能です。

アンチグリッチ回路は、トリガ信号のデジタル積分を行います。インパルス性ノイズを取り除くことに有効です。

アンチチャタリング回路は、トリガの誤動作を防止するためにエッジを受け付けない時間を設定します。不安定な論理状態やスイッチチャタリングを取り除くことに有効です。



カメラのアンチグリッチとアンチチャタリングに関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.13.1. `get_anti_glitch_min_max`

カメラのデジタル入力信号の積分時間（絶対値）の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_anti_glitch_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値（単位：マイクロ秒）

max : 最大値（単位：マイクロ秒）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AntiGlitch レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.13.2. `get_anti_glitch`

カメラのデジタル入力信号の積分時間（絶対値）設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_anti_glitch (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : デジタル入力信号の積分時間（絶対値）（単位：マイクロ秒）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AntiGlitch レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.13.3. set_anti_glitch

カメラのデジタル入力信号の積分時間（絶対値）を設定します。

[構文]

```
pytelicam.CameraControl.set_anti_glitch(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	設定するデジタル入力信号の積分時間（絶対値）です。 （単位：マイクロ秒）

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AntiGlitch レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.13.4. `get_anti_chattering_min_max`

カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_anti_chattering_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値（単位：マイクロ秒）

max : 最大値（単位：マイクロ秒）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AntiChattering レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.13.5. `get_anti_chattering`

カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_anti_chattering (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : デジタル入力信号のエッジを受け付けない時間（絶対値）（単位：マイクロ秒）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

AntiChattering レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.13.6. set_anti_chattering

カメラのデジタル入力信号のエッジを受け付けない時間（絶対値）を設定します。

[構文]

```
pytelicam.CameraControl.set_anti_chattering(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	設定するデジタル入力信号のエッジを受け付けない時間（絶対値）です。（単位：マイクロ秒）

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

AntiChattering レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

本関数では単位が マイクロ秒になっていることに注意してください。

7.6.14. TimerControl

カメラの TimerControl 機能（Timer0Active 信号）の制御を行います。

カメラはタイマーを使用して TimerActive 信号を作成します。



本章のメソッドは Timer0Active 信号を制御します。（BG、BU シリーズはタイマーを 1 本のみ搭載しています。）

カメラの TimerControl 機能（Timer0Active 信号）に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.14.1. get_timer_duration_min_max

カメラの Timer0Active 信号の幅の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_timer_duration_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値（単位：マイクロ秒）
max : 最大値（単位：マイクロ秒）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TimerDuration レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.2. get_timer_duration

カメラの Timer0Active 信号の幅を取得します。

[構文]

```
pytelicam.CameraControl.get_timer_duration (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : Timer0Active 信号の幅 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TimerDuration レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.3. set_timer_duration

カメラの Timer0Active 信号の幅を設定します。

[構文]

```
pytelicam.CameraControl.set_timer_duration(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	Timer0Active 信号の幅です。(単位: マイクロ秒)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TimerDuration レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.4. get_timer_delay_min_max

カメラの Timer0Active 信号の遅延量の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_timer_delay_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: マイクロ秒)

max : 最大値 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TimerDelay レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.5. get_timer_delay

カメラの Timer0Active 信号の遅延量を取得します。

[構文]

```
pytelicam.CameraControl.get_timer_delay (self)
```

[戻り値]

tuple 型のデータ (status, microseconds)

status : 実行結果を表すステータスコード

microseconds : Timer0Active 信号の遅延量 (単位: マイクロ秒)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TimerDelay レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.6. set_timer_delay

カメラの Timer0Active 信号の遅延量を設定します。

[構文]

```
pytelicam.CameraControl.set_timer_delay(self, microseconds)
```

[パラメータ]

パラメータ	内 容
microseconds (double)	Timer0Active 信号の遅延量です。(単位：マイクロ秒)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TimerDelay レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.7. get_timer_trigger_source

カメラの Timer0Active 信号の基準信号を取得します。

[構文]

```
pytelicam.CameraControl.get_timer_trigger_source(self)
```

[戻り値]

tuple 型のデータ (status, source)

status : 実行結果を表すステータスコード

source : Timer0Active 信号の基準信号

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraTimerTriggerSource](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

TimerTriggerSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.14.8. set_timer_trigger_source

カメラの Timer0Active 信号の基準信号を設定する。

[構文]

```
pytelicam.CameraControl.set_timer_trigger_source(self, source)
```

[パラメータ]

パラメータ	内 容
source (pytelicam.CameraTimerTriggerSource)	Timer0Active 信号の基準信号です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

TimerTriggerSource レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラにより、設定できる基準信号種類が異なります。

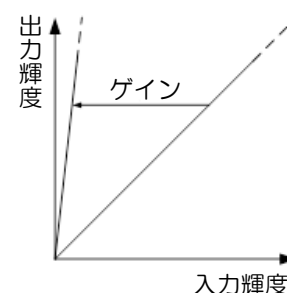
カメラの TimerConrtol 機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.15. Gain

カメラのゲイン調整機能の制御を行います。

ゲインを設定することで、映像輝度の倍率を変更することができます。制御方式としてマニュアルゲイン（MANUAL）と自動ゲイン制御（AGC）が利用可能です。AGC では被写体の明るさに応じてゲインを自動で調整します。

カメラのゲイン調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.15.1. get_gain_min_max

カメラのゲインの最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_gain_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値 (単位: dB または 倍)
max : 最大値 (単位: dB または 倍)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Gain レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.15.2. get_gain

カメラのゲインを取得します。

[構文]

```
pytelicam.CameraControl.get_gain (self)
```

[戻り値]

tuple 型のデータ (status, gain)

status : 実行結果を表すステータスコード

gain : ゲイン (単位: dB または 倍)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Gain レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.15.3. set_gain

カメラのゲインを設定する。

[構文]

```
pytelicam.CameraControl.set_gain(self, gain)
```

[パラメータ]

パラメータ	内 容
gain (double)	ゲインです。(単位: dB または 倍)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

Gain レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.15.4. get_gain_auto

カメラの AGC (Automatic gain control) 動作モードを取得します。

[構文]

```
pytelicam.CameraControl.get_gain_auto(self)
```

[戻り値]

tuple 型のデータ (status, value)
status : 実行結果を表すステータスコード
value : AGC 動作モード

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraGainAuto](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Gain または GainAuto レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.15.5. set_gain_auto

カメラの AGC (Automatic gain control) 動作モードを設定します。

[構文]

```
pytelicam.CameraControl.set_gain_auto(self, value)
```

[パラメータ]

パラメータ	内 容
value (pytelicam.CameraGainAuto)	AGC 動作モードです。

[戻り値]

実行結果を表すステータスコード。
主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

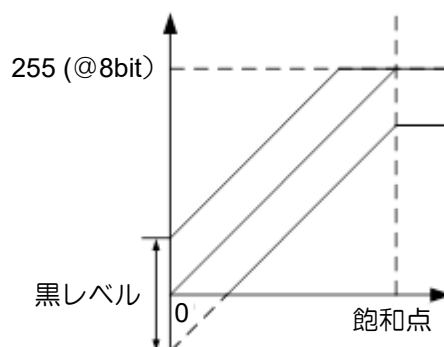
[備考]

Gain または GainAuto レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.16. BlackLevel

カメラの黒レベル調整機能の制御を行います。

カメラの黒レベル調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.16.1. get_black_level_min_max

カメラの映像の黒レベルの最小値と最大値を取得します。

【構文】

```
pytelicam.CameraControl.get_black_level_min_max (self)
```

【戻り値】

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: %)

max : 最大値 (単位: %)

【戻り値の型】

tuple([pytelicam.CamApiStatus](#), double, double)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

【備考】

BlackLevel レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.16.2. get_black_level

カメラの映像の黒レベルを取得します。

[構文]

```
pytelicam.CameraControl.get_black_level (self)
```

[戻り値]

tuple 型のデータ (status, black_level)
status : 実行結果を表すステータスコード
black_level : 黒レベル (単位 : %)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

BlackLevel レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.16.3. set_black_level

カメラの映像の黒レベルを設定します。

[構文]

```
pytelicam.CameraControl.set_black_level(self, black_level)
```

[パラメータ]

パラメータ	内 容
black_level (double)	黒レベルです。(単位 : %)

[戻り値]

実行結果を表すステータスコード。
主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

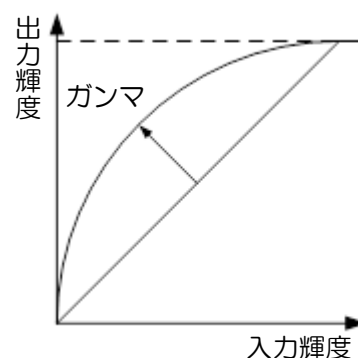
[備考]

BlackLevel レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.17. Gamma

カメラのガンマ補正機能の制御を行います。

カメラのガンマ補正機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.17.1. get_gamma_min_max

カメラのガンマ補正值の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_gamma_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Gamma レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.17.2. get_gamma

カメラのガンマ補正値を取得します。

[構文]

```
pytelicam.CameraControl.get_gamma (self)
```

[戻り値]

tuple 型のデータ (status, gamma)

status : 実行結果を表すステータスコード

gamma : ガンマ補正値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Gamma レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.17.3. set_gamma

カメラのガンマ補正値を設定します。

[構文]

```
pytelicam.CameraControl.set_gamma(self, gamma)
```

[パラメータ]

パラメータ	内 容
gamma (double)	ガンマ補正値です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

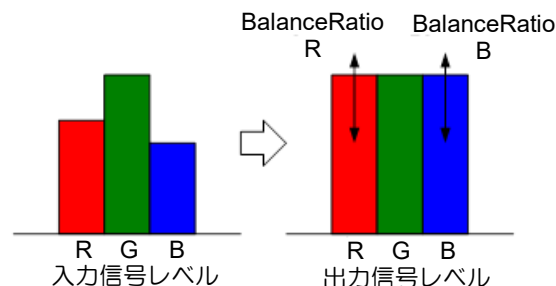
Gamma レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.18. WhiteBalance

カメラのホワイトバランス調整機能の制御を行います。

この章の関数はカラーカメラで使用できます。

カメラはGコンポーネントを基準とした R と B コンポーネントのゲインの比率を手動または自動で調整することによりホワイトバランスを制御しています。



カメラのホワイトバランス調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.18.1. get_balance_ratio_min_max

カメラのホワイトバランスゲイン（倍率）の最小値と最大値を取得します。

【構文】

```
pytelicam.CameraControl.get_balance_ratio_selector (self, selector)
```

【パラメータ】

パラメータ	内 容
selector (pytelicam. CameraBalanceRatioSelector)	ホワイトバランスゲイン設定の対象となる要素です。

【戻り値】

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

【戻り値の型】

tuple([pytelicam.CamApiStatus](#), double, double)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

【備考】

WhiteBalance* レジスタ（または BalanceRatioSelector / BalanceRatio ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.18.2. get_balance_ratio

カメラのホワイトバランスゲイン（倍率）を取得します。

[構文]

```
pytelicam.CameraControl.get_balance_ratio(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraBalanceRatioSelector)	ホワイトバランスゲイン設定の対象となる要素です。

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ホワイトバランスゲイン（倍率）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

WhiteBalance* レジスタ（または BalanceRatioSelector / BalanceRatio ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.18.3. set_balance_ratio

カメラのホワイトバランスゲイン（倍率）を設定します。

[構文]

```
pytelicam.CameraControl.set_balance_ratio(self, selector, value)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraBalanceRatioSelector)	ホワイトバランスゲイン設定の対象となる要素です。
value (double)	設定する信号種類です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

WhiteBalance* レジスタ（または BalanceRatioSelector / BalanceRatio ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.18.4. get_balance_white_auto

カメラのホワイトバランスゲイン自動調整モードを取得します。

[構文]

```
pytelicam.CameraControl.get_balance_white_auto(self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ホワイトバランスゲイン自動調整モード

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraBalanceWhiteAuto](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

WhiteBalance* レジスタ（または BalanceWhiteAuto ノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.18.5. set_balance_white_auto

カメラのホワイトバランスゲイン自動調整モードを設定します。

[構文]

```
pytelicam.CameraControl.set_balance_white_auto(self, value)
```

[パラメータ]

パラメータ	内 容
value (pytelicam.CameraBalanceWhiteAuto)	ホワイトバランスゲイン自動調整モードです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

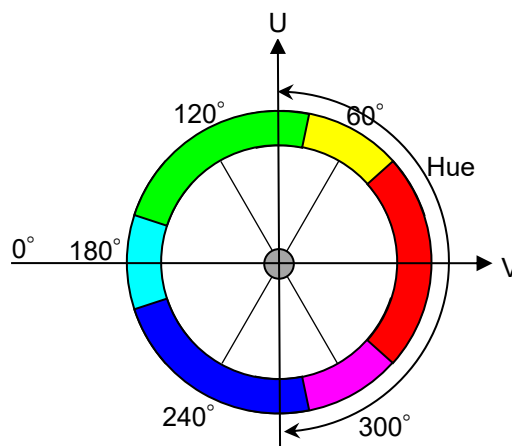
ホワイトバランス調整機能（BalanceWhiteAuto レジスタ）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.19. Hue

カメラの色相調整機能の制御を行います。

この章の関数はカラーカメラで使用できます。

カメラの色相調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.19.1. get_hue_min_max

カメラの色相の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_hue_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: °)

max : 最大値 (単位: °)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Hue レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.19.2. get_hue

カメラの色相の設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_hue(self)
```

[戻り値]

tuple 型のデータ (status, hue)

status : 実行結果を表すステータスコード

hue : カメラの色相の設定値 (単位: °)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Hue レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.19.3. set_hue

カメラの色相を設定します。

[構文]

```
pytelicam.CameraControl.set_hue(self, hue)
```

[パラメータ]

パラメータ	内 容
hue (double)	カメラの色相です。(単位: °)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

Hue レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

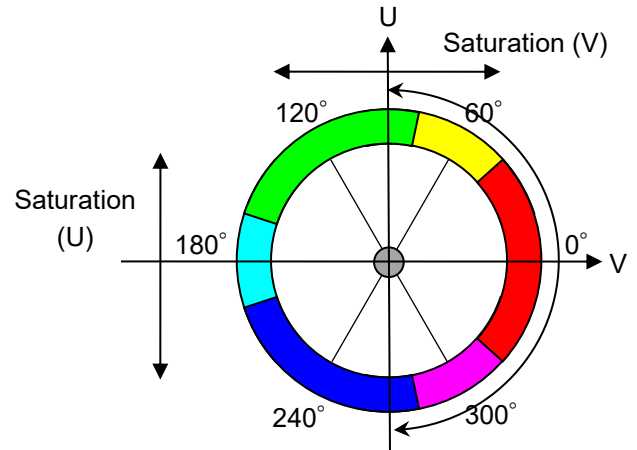
7.6.20. Saturation

カメラの彩度調整機能の制御を行います。

この章のメソッドはカラーカメラで使用できます。

対象となる要素 (U/V) を選択して [倍] 単位で設定するカメラと、U/V 要素共通で [%] 単位で設定するカメラがあります。

カメラの彩度調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.20.1. get_saturation_min_max

カメラの彩度の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_saturation_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値 (単位: % または 倍)

max : 最大値 (単位: % または 倍)

[戻り値の型]

```
tuple(pytelicam.CamApiStatus, double, double)
```

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Saturation レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラの彩度を [倍] 単位で設定するカメラと、[%] 単位で U/V 要素共通で設定するカメラがあります。

カメラの彩度調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.20.2. get_saturation

カメラの彩度の設定値を取得します。

[構文]

```
pytelicam.CameraControl.get_saturation(self, selector)
pytelicam.CameraControl.get_saturation(self)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraSaturationSelector)	彩度設定の対象となる要素です。

[戻り値]

tuple 型のデータ (status, saturation)

status : 実行結果を表すステータスコード

saturation : カメラの彩度の設定値 (単位: % または 倍)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Saturation レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

対象となる要素 (U/V) を選択して [倍] 単位で設定するカメラと、[%] 単位で U/V 要素共通で設定するカメラがあります。

U/V 要素共通で設定するカメラの場合は、引数 *selector* の無いメソッドを使用してください。

対象となる要素 (U/V) を選択して設定するカメラの場合は、引数 *selector* により要素 (U/V) を選択してください。

カメラの彩度調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.20.3. set_saturation

カメラの彩度を設定します。

[構文]

```
pytelicam.CameraControl.set_saturation(self, selector, saturation)
pytelicam.CameraControl.set_saturation(self, saturation)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraSaturationSelector)	彩度設定の対象となる要素です。
saturation (double)	カメラの彩度です。(単位： % または 倍)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

Saturation レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

対象となる要素（U/V）を選択して [倍] 単位で設定するカメラと、 [%] 単位で U/V 要素共通で設定するカメラがあります。

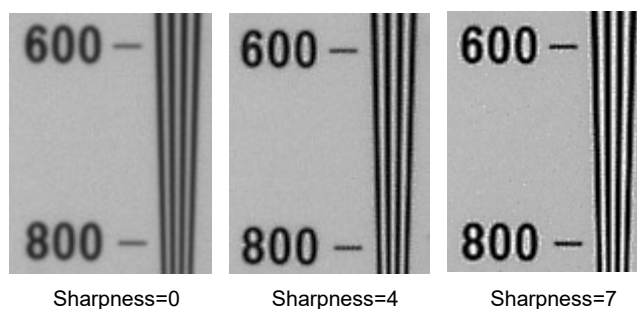
U/V 要素共通で設定するカメラの場合は、引数 *selector* の無いメソッドを使用してください。

対象となる要素（U/V）を選択して設定するカメラの場合は、引数 *selector* により要素（U/V）を選択してください。

カメラの彩度調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.21. Sharpness

カメラの画像のエッジ強度を調整します。



カメラの画像のエッジ強度の説明は、使用するカメラの取扱説明書をご覧ください。

7.6.21.1. get_sharpness_min_max

カメラの画像のエッジ強度（sharpness）の最小値と最大値を取得します。

【構文】

```
pytelicam.CameraControl.get_sharpness_min_max (self)
```

【戻り値】

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値

max : 最大値

【戻り値の型】

tuple([pytelicam.CamApiStatus](#), int, int)

【例外】

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

【備考】

Sharpness レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.21.2. get_sharpness

カメラの画像のエッジ強度（sharpness）を取得します。

[構文]

```
pytelicam.CameraControl.get_sharpness (self)
```

[戻り値]

tuple 型のデータ（status, sharpness）

status : 実行結果を表すステータスコード

sharpness : 画像のエッジ強度の設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Sharpness レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.21.3. set_sharpness

カメラの画像のエッジ強度（sharpness）を設定します。

[構文]

```
pytelicam.CameraControl.set_sharpness(self, sharpness)
```

[パラメータ]

パラメータ	内 容
sharpness (int)	画像のエッジ強度です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

エッジ強度調整機能（Sharpness レジスタ）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.22. ColorCorrectionMatrix

カメラの色補正マトリックス調整機能の制御を行います。

この章のメソッドはカラーカメラで使用できます。

原画素値 (R, G, B) と色補正後の画素値 (R', G', B') の関係は以下の式の関係になります。

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} 1 & -mask_rg & -mask_rb \\ -mask_gr & 1 & -mask_gb \\ -mask_br & -mask_bg & 1 \end{pmatrix} \begin{pmatrix} R & (G-R) & (B-R) \\ (R-G) & G & (B-G) \\ (R-B) & (G-B) & B \end{pmatrix}$$

$$R' = (1 - mask_rg - mask_rb) \times R + mask_rg \times G + mask_rb \times B$$

$$G' = mask_gr \times R + (1 - mask_gr - mask_gb) \times G + mask_gb \times B$$

$$B' = mask_br \times R + mask_bg \times G + (1 - mask_br - mask_bg) \times B$$

GenApi ライブラリでは"SelectorI"と"SelectorJ"の2個のセレクトラを使用して行列の要素を指定しています。"SelectorI"と"SelectorJ"と色補正マトリックスの要素は以下の関係になります。

	SelectorJ = R	SelectorJ = G	SelectorJ = B
SelectorI = R		mask_rg	mask_rb
SelectorI = G	mask_gr		mask_gb
SelectorI = B	mask_br	mask_bg	

本章のメソッドは、"SelectorI"の値と"SelectorJ"の値を1個の列挙子にまとめた列挙型 [CameraColorCorrectionMatrixSelector](#) を行列の要素指定に使用しています。

カメラの色補正マトリックス調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.22.1. get_color_correction_matrix_min_max

カメラの色補正マトリックスの係数の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_color_correction_matrix_min_max (self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraColorCorrectionMatrixSelector)	色補正マトリックスの要素です。

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード

min : 最小値

max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double, double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Masking* レジスタ、または ColorCorrectionMatrix / ColorCorrectionMatrixSelectorI / ColorCorrectionMatrixSelectorJ レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.22.2. get_color_correction_matrix

カメラの色補正マトリックスの係数を取得します。

[構文]

```
pytelicam.CameraControl.get_color_correction_matrix(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraColorCorrectionMatrixSelector)	色補正マトリックスの要素です。

[戻り値]

tuple 型のデータ (status, matrix)

status : 実行結果を表すステータスコード

matrix : 色補正マトリックスの係数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), double)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

Saturation レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

対象となる要素（U/V）を選択して [倍] 単位で設定するカメラと、 [%] 単位で U/V 要素共通で設定するカメラがあります。

7.6.22.3. set_color_correction_matrix

カメラの色補正マトリックスの係数を設定します。

[構文]

```
pytelicam.CameraControl.set_color_correction_matrix(self, selector, matrix)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraColorCorrectionMatrixSelector)	色補正マトリックスの要素です。
matrix (double)	色補正マトリックスの係数です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

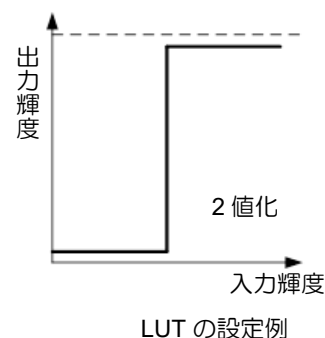
[備考]

Masking* レジスタ、または ColorCorrectionMatrix / ColorCorrectionMatrixSelectorI / ColorCorrectionMatrixSelectorJ レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.23. LUT Control

カメラの LUT（ルックアップテーブル）調整機能の制御を行います。

カメラの LUT 調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。



7.6.23.1. get_lut_enable

カメラの LUT モード（有効 / 無効）を取得します。

[構文]

```
pytelicam.CameraControl.get_lut_enable (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : LUT モード（有効 / 無効）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LUTEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.23.2. set_lut_enable

カメラの LUT モード（有効 / 無効）を設定します。

[構文]

```
pytelicam.CameraControl.set_lut_enable(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	LUT モード（有効 / 無効）です。 true とき有効、false のとき無効です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LUTEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.23.3. get_lut_value

カメラの LUT の出力値を取得します。

[構文]

```
pytelicam.CameraControl.get_lut_value (self, index)
```

[パラメータ]

パラメータ	内 容
index (int)	LUT の入力値です。 (設定範囲：0 ～ 1023 または 0 ～ 4095)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : LUT の出力値 (取得範囲：0 ～ 1023 または 0 ～ 4095)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

LUTValueAll、または LUTIndex と LUTValue レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

LUT の入力値と出力値の範囲は、カメラにより異なります。

カメラの LUT 調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.23.4. set_lut_value

カメラの LUT の出力値を設定します。

[構文]

```
pytelicam.CameraControl.set_lut_value(self, index, value)
```

[パラメータ]

パラメータ	内 容
index (int)	LUT の入力値です。(設定範囲: 0 ~ 1023 または 0 ~ 4095)
value (int)	LUT の出力値です。(設定範囲: 0 ~ 1023 または 0 ~ 4095)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

LUTValueAll、または LUTIndex と LUTValue レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

LUT の入力値と出力値の範囲は、カメラにより異なります。

カメラの LUT 調整機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.24. UserSetControl

カメラのユーザー設定機能の制御を行います。
カメラのユーザー設定機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.24.1. execute_user_set_load

カメラに実装されている不揮発性メモリ（ユーザーメモリ）から、設定パラメータをカメラにロードします。

[構文]

```
pytelicam.CameraControl.execute_user_set_load(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraUserSetSelector)	ロードするユーザー設定チャンネルです。

[戻り値]

実行結果を表すステータスコード。
主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserSetSelector と UserSetCommand、または UserSetLoad レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

ロードされるパラメータは、使用するカメラの取扱説明書をご覧ください。

7.6.24.2. execute_user_set_save

現在カメラに設定されているパラメータを、カメラに実装されている不揮発性メモリ（ユーザーメモリ）にセーブします。

[構文]

```
pytelicam.CameraControl.execute_user_set_save(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraUserSetSelector)	セーブするユーザー設定チャンネルです。 CameraUserSetSelector.Default は指定できません。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserSetSelector と UserSetCommand、または UserSetSave レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

セーブされるパラメータ、および UserSetSave と UserSetQuickSave の違いは使用するカメラの取扱説明書をご覧ください。

7.6.24.3. execute_user_set_quick_save

現在カメラに設定されているパラメータを、カメラに実装されている揮発性メモリ（ユーザーメモリ）にセーブします。

[構文]

```
pytelicam.CameraControl.execute_user_set_quick_save(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraUserSetSelector)	セーブするユーザー設定チャンネルです。 CameraUserSetSelector.Default は指定できません。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserSetSelector と UserSetCommand、または UserSetQuickSave レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

[execute_user_set_save\(\)](#) に比べ高速に処理できますが、揮発性メモリ（カメラ内部 RAM）にセーブされるため、カメラの電源が OFF されるとセーブしたデータは消えてしまいます。

セーブされるパラメータ、および UserSetSave と UserSetQuickSave の違いはカメラの取扱説明書をご覧ください。

7.6.24.4. get_user_set_default

カメラの起動時にロードするユーザー設定チャンネルを読み出します。

[構文]

```
pytelicam.CameraControl.get_user_set_default (self)
```

[戻り値]

tuple 型のデータ (status, selector)

status : 実行結果を表すステータスコード

selector : ユーザー設定チャンネル

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraUserSetSelector](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

UserSetDefault レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.24.5. set_user_set_default

カメラの起動時にロードするユーザー設定チャンネルを設定します。

[構文]

```
pytelicam.CameraControl.set_user_set_quick_save(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraUserSetSelector)	セーブするユーザー設定チャンネルです。 CameraUserSetSelector.Default は指定できません。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserSetSelector と UserSetCommand、または UserSetQuickSave レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

カメラのファームウェアバージョンにより、不揮発性メモリにセーブされるカメラとセーブされないカメラがあります。不揮発性メモリにセーブされないカメラの場合は、[execute_user_set_save_and_set_default\(\)](#) により、不揮発性メモリへのセーブも同時に行ってください。

カメラのユーザー設定機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.24.6. execute_user_set_save_and_set_default

現在カメラに設定されているパラメータを、カメラに実装されている不揮発性メモリ（ユーザーメモリ）にセーブし、カメラ起動時のユーザー設定

[構文]

```
pytelicam.CameraControl.execute_user_set_save_and_set_default(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraUserSetSelector)	セーブするユーザー設定チャンネルです。 CameraUserSetSelector.Default は指定できません。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserSetSelector と UserSetCommand、または UserSetSave レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

selector に CameraUserSetSelector.Default 以外を指定したときは、指定されたユーザー設定チャンネルに現在のパラメータがセーブされます。カメラ起動時は指定されたユーザー設定チャンネルのパラメータで起動されます。

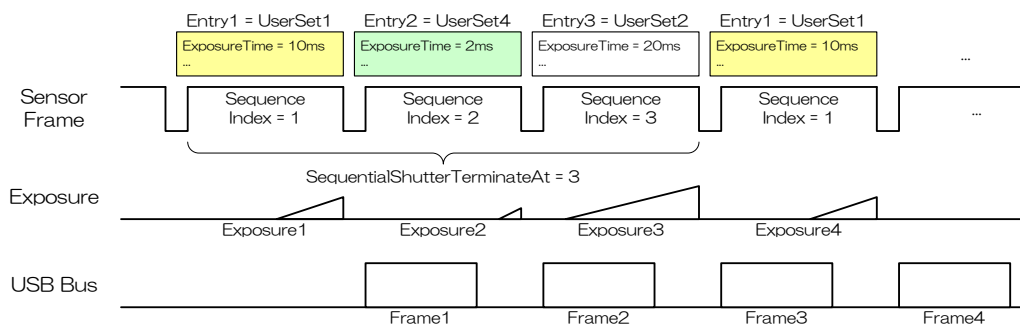
selector に [CameraUserSetSelector](#).Default を指定したときは、工場出荷時設定がロードされ、現在のパラメータのセーブは行われません。カメラ起動時は工場出荷時設定のパラメータで起動されます。

セーブされるパラメータは、カメラの取扱説明書をご覧ください。

7.6.25. SequentialShutterControl

カメラのシーケンシャルシャッター機能の設定を行います。

シーケンシャルシャッターは露光のたびに画像取得パラメータを **UserSet** に保存されている値に順番に切り替えて撮像する機能です。



カメラのシーケンシャルシャッター機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.25.1. get_sequential_shutter_enable

カメラのシーケンシャルシャッターモード（有効 / 無効）を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_enable (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : シーケンシャルシャッターモード（有効 / 無効）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.2. set_sequential_shutter_enable

カメラのシーケンシャルシャッターモード（有効 / 無効）を設定します。

[構文]

```
pytelicam.CameraControl.set_sequential_shutter_enable(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	シーケンシャルシャッターモード（有効 / 無効）です。 true とき有効、false のとき無効です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.3. get_sequential_shutter_terminate_at_min_max

カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_terminate_at_min_max (self)
```

[戻り値]

tuple 型のデータ（status, min, max）

status : 実行結果を表すステータスコード

min : 最小値

max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterTerminateAt レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.4. get_sequential_shutter_terminate_at

カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_terminate_at(self)
```

[戻り値]

tuple 型のデータ (status, value)
status : 実行結果を表すステータスコード
value : Sequence の繰り返しを行うインデックス数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterTerminateAt レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.5. set_sequential_shutter_terminate_at

カメラのシーケンシャルシャッター機能の Sequence の繰り返しを行うインデックス数を設定します。

[構文]

```
pytelicam.CameraControl.set_sequential_shutter_terminate_at(self, value)
```

[パラメータ]

パラメータ	内 容
value (int)	Sequence の繰り返しを行うインデックス数です。

[戻り値]

実行結果を表すステータスコード。
主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterTerminateAt レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.6. get_sequential_shutter_index_min_max

カメラのシーケンシャルシャッター機能の登録を行うシーケンス番号の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_index_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterSequenceTable または SequentialShutterIndex レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.7. get_sequential_shutter_entry_min_max

カメラのシーケンシャルシャッター機能のシーケンスに登録するユーザー設定チャンネル (UserSet 番号) の最小値と最大値を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_entry_min_max (self)
```

[戻り値]

tuple 型のデータ (status, min, max)

status : 実行結果を表すステータスコード
min : 最小値
max : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int, int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterSequenceTable または SequentialShutterEntry レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.8. get_sequential_shutter_entry

カメラのシーケンシャルシャッター機能のシーケンスに登録されているユーザー設定チャンネル（UserSet 番号）を取得します。

[構文]

```
pytelicam.CameraControl.get_sequential_shutter_entry (self, index)
```

[パラメータ]

パラメータ	内 容
index (int)	シーケンス番号です。

[戻り値]

tuple 型のデータ (status, entry)

status : 実行結果を表すステータスコード

entry : ユーザー設定チャンネル（UserSet 番号）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterSequenceTable または SequentialShutterEntry レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.25.9. set_sequential_shutter_entry

カメラのシーケンシャルシャッター機能のユースケースにユーザー設定チャンネル（UserSet 番号）を登録します。

[構文]

```
pytelicam.CameraControl.set_sequential_shutter_entry(self, index, entry)
```

[パラメータ]

パラメータ	内 容
index (int)	登録するシーケンス番号です。
entry (int)	登録するユーザー設定チャンネル（UserSet 番号）です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

SequentialShutterSequenceTable または SequentialShutterEntry レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.26. UserDefinedName

カメラのユーザー定義情報（UserDefinedName または、DeviceUserID レジスタ）の制御を行います。

UserDefinedName または DeviceUserID レジスタは、カメラ内部の不揮発性メモリに任意の文字列を保存することができるレジスタ（メモリ）です。複数台カメラを使用する場合、カメラを特定する手段として利用することができます。

ユーザー定義情報（UserDefinedName または DeviceUserID レジスタ）に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.26.1. get_user_defined_name

カメラのユーザー定義情報を取得します。

[構文]

```
pytelicam.CameraControl.get_user_defined_name (self)
```

[戻り値]

tuple 型のデータ (status, name)

status : 実行結果を表すステータスコード

name : ユーザー定義情報

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), string)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

UserDefinedName または DeviceUserID レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

設定できる文字数は、カメラにより異なります。

GigE カメラは最大 16 文字(16byte)、USB3 カメラは最大 64 文字(64byte)です。

TeliCamDNetAPI では、終端に null 文字を付与してカメラに設定するため、name で指定できる文字数は、上記の最大文字数 - 1 となります。

7.6.26.2. set_user_defined_name

カメラのユーザー定義情報を設定します。

[構文]

```
pytelicam.CameraControl.set_user_defined_name(self, name)
```

[パラメータ]

パラメータ	内 容
name (string)	ユーザー定義情報です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

UserDefinedName または DeviceUserID レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

設定できる文字数は、カメラにより異なります。

GigE カメラは最大 16 文字(16byte)、USB3 カメラは最大 64 文字(64byte)です。

TeliCamDNetAPI では、終端に null 文字を付与してカメラに設定するため、name で指定できる文字数は、上記の最大文字数 - 1 となります。

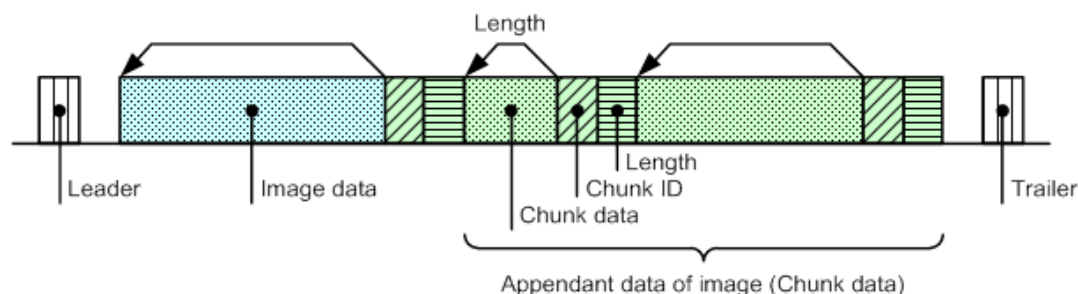
7.6.27. Chunk

カメラのチャンク機能の設定を行います。

チャンクデータとは画像データ毎に付加されたタグ情報を指します。

このタグ情報はアプリケーションがデータのペイロードを解析して様々な要素を抽出・識別できるようにするものです。

有効化されたチャンクデータの内容が多くなると、そのフレーム長は長くなります。



カメラのチャンク機能に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

GenICam を使用してチャンクデータを取得するときは、ペイロードデータが格納されているバッファを GenICam のチャンクアダプタにアタッチする必要があります。 詳細は、[CameraStream クラスの chunk_attach_buffer\(\)](#) の説明をご覧ください。

7.6.27.1. get_chunk_mode_active

カメラのチャンク機能の有効状態を取得します。

[構文]

```
pytelicam.CameraControl.get_chunk_mode_active (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 有効状態

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ChunkModeActive レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.2. set_chunk_mode_active

カメラのチャンク機能の有効／無効を設定します。

[構文]

```
pytelicam.CameraControl.set_chunk_mode_active(self, value)
```

[パラメータ]

パラメータ	内 容
value (bool)	設定する有効状態の値です。 true とき有効、false のとき無効です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ChunkModeActive レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.3. get_chunk_enable

カメラのチャンクデータの有効状態を取得する。

[構文]

```
pytelicam.CameraControl.get_chunk_enable(self, selector)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraChunkSelector)	取得の対象となるチャンクデータです。

[戻り値]

tuple 型のデータ (status, value)
status : 実行結果を表すステータスコード
value : 取得した有効状態

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ChunkEnableOf*、または ChunkSelector と ChunkEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.4. set_chunk_enable

カメラのチャンクデータの有効状態を設定します。

[構文]

```
pytelicam.CameraControl.set_chunk_enable(self, selector, value)
```

[パラメータ]

パラメータ	内 容
selector (pytelicam.CameraChunkSelector)	設定の対象となるチャンクデータです。
value (double)	設定する有効状態の値です。 true とき有効、false のとき無効です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ChunkEnableOf*、または ChunkSelector と ChunkEnable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.5. get_chunk_user_area_length

カメラの ChunkUserAreaTable の長さを取得します。

[構文]

```
pytelicam.CameraControl.get_chunk_user_area_length (self)
```

[戻り値]

tuple 型のデータ (status, length)

status : 実行結果を表すステータスコード

length : ChunkUserAreaTable の長さ (byte 単位)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ChunkUserArea または ChunkUserAreaTable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.6. get_chunk_user_area_table

カメラの ChunkUserAreaTable に設定されているデータを取得します。

[構文]

```
pytelicam.CameraControl.get_chunk_user_area_table (self)
```

[戻り値]

tuple 型のデータ (status, data)

status : 実行結果を表すステータスコード

data : 取得したデータ

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), string)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

ChunkUserArea または ChunkUserAreaTable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.27.7. set_chunk_user_area_table

カメラの ChunkUserAreaTable にデータを設定します。

[構文]

```
pytelicam.CameraControl.set_chunk_user_area_table(self, data)
```

[パラメータ]

パラメータ	内 容
data (string)	設定する string 型のデータです。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

ChunkUserArea または ChunkUserAreaTable レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.28. FrameSynchronization

カメラのフレーム同期の制御を行います。
フレーム同期は、USB3 カメラのみ有効です。

カメラのフレーム同期に関する詳しい説明は、使用するカメラの取扱説明書をご覧ください。

7.6.28.1. get_frame_synchronization

カメラのチャंक機能の有効状態を取得します。

[構文]

```
pytelicam.CameraControl.get_frame_synchronization (self)
```

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : Sequence の繰り返しを行うインデックス数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.CameraFrameSynchronization](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

FrameSynchronization レジスタ (またはノード) が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.6.28.2. set_frame_synchronization

カメラのチャック機能の有効／無効を設定します。

[構文]

```
pytelicam.CameraControl.set_frame_synchronization(self, value)
```

[パラメータ]

パラメータ	内 容
value (pytelicam.CameraFrameSynchronization)	フレーム同期制御方法を格納する変数の参照です。

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

FrameSynchronization レジスタ（またはノード）が実装されていないカメラで実行するとエラーステータスがリターンされます。

7.7. GenApiWrapper クラス (カメラ制御用サブセット)

カメラの各機能を制御するための GenICam GenApi モジュールをラップしたクラスです。

殆どのカメラは GenICam 規格に準拠したカメラ記述ファイル（XML ファイル）を持っています。本クラスは XML ファイルの情報と GenICam GenApi のモジュールを使用して、レジスタアドレス指定ではなくフィーチャ名を指定してカメラの制御を行うクラスです。

GenICam に関する詳細情報は <https://www.emva.org/standards-technology/genicam/> をご覧ください。

GenApiWrapper クラスのオブジェクトは [CameraDevice](#) クラスで管理されており、ユーザーアプリケーションで オブジェクトの作成および破棄 (del) を行う必要はありません。

[構文]

```
pytelicam.GenApiWrapper
```

[関数]

	名 称	内 容
INode 型関数		
⇒	get_node_type	ノードの種類を取得します。
⇒	get_access_mode	ノードのアクセスモードを取得します。
⇒	get_visibility	ノードの可視性（機能にアクセスできるユーザーの推奨レベル）を取得します。
⇒	get_caching_mode	ノードのキャッシュ設定を取得します。
⇒	get_description	ノードの説明を取得します。
⇒	get_tool_tip	ノードのツールチップ（簡単な説明）を取得します。
⇒	get_representation	ノードの推奨表現を取得します。
⇒	get_unit	ノードの単位名を取得します。
ICategory 型メソッド		
⇒	get_num_of_features	ICategory 型ノードが持つ子 Feature ノードの数を取得します。
⇒	get_feature_name	ICategory 型ノードが持つ子 Feature ノードの名前を取得します。
⇒	get_available_feature_names	ICategory 型ノードが持つ有効な子 Feature ノードの名前のタプル（リスト）を取得します。
共通関数		
⇒	get_feature_value	Integer 型、IFloat 型、Boolean 型、Enumeration 型、String 型ノードの現在の値を文字列として取得します。
⇒	set_feature_value	Integer 型、IFloat 型、Boolean 型、Enumeration 型、String 型ノードの値を文字列で設定します。
Integer 型関数		
⇒	get_int_min	Integer 型ノードの現在有効な最小値を取得します。
⇒	get_int_max	Integer 型ノードの現在有効な最大値を取得します。
⇒	get_int_inc	Integer 型ノードの増加量を取得します。
⇒	get_int_value	Integer 型ノードの値を取得します。
⇒	set_int_value	Integer 型ノードの値を設定します。
IFloat 型関数		

	名 称	内 容
⇒	get_float_min	IFloat 型ノードの現在有効な最小値を取得します。
⇒	get_float_max	IFloat 型ノードの現在有効な最大値を取得します。
⇒	get_float_has_inc	IFloat 型ノードの増加量を取得・設定する機能が実装されているかどうかを取得します。
⇒	get_float_inc	IFloat 型ノードの増加量を取得します。
⇒	get_float_display_notation	IFloat 型ノードの文字表示方法を取得します。
⇒	get_float_display_precision	IFloat 型ノードの値を文字に変換する時に使用する小数点以下の精度を取得します。
⇒	get_float_value	IFloat 型ノードの値を取得します。
⇒	set_float_value	IFloat 型ノードの値を設定します。
IBoolean 型関数		
⇒	get_bool_value	IBoolean 型ノードの値を取得します。
⇒	set_bool_value	IBoolean 型ノードの値を設定します。
IEnumeration 型、IEnumEntry 関数		
⇒	get_enum_int_value	IEnumeration 型ノードの値を整数値として取得します。
⇒	set_enum_int_value	IEnumeration 型ノードの値を整数値で設定します。
⇒	get_enum_str_value	IEnumeration 型ノードの値を文字列として取得します。
⇒	set_enum_str_value	IEnumeration 型ノードの値を文字列で設定します。
⇒	get_available_enum_entry_names	IEnumeration 型ノードが持つ有効な Entry ノードの名前のタプル（リスト）を取得します。
⇒	get_num_of_enum_entries	IEnumeration 型ノードのエントリー数を取得します。
⇒	get_enum_entry_access_mode	IEnumeration 型ノードが保有する IEnumEntry 型ノードリスト内のインデックスを指定し、IEnumEntry 型ノードのアクセスモードを取得します。
⇒	get_enum_entry_int_value	IEnumEntry 型ノードの値を整数値で取得します。
⇒	get_enum_entry_str_value	IEnumEntry 型ノードの値を文字列で取得します。
ICommand 型関数		
⇒	execute_command	ICommand 型ノードのコマンドを実行します。
⇒	get_command_is_done	ICommand 型ノードのコマンド実行状態を取得します。
IString 型関数		
⇒	get_str_value	IString 型ノードの値（文字列）を取得します。
⇒	set_str_value	IString 型ノードの値（文字列）を設定します。
その他関数		
⇒	get_access_module	GenApiWrapper クラスのメソッドでアクセスするモジュール（xml ファイル）の設定を取得します。
⇒	set_access_module	GenApiWrapper クラスのメソッドでアクセスするモジュール（xml ファイル）を設定します。

7.7.1. INode 型関数

7.7.1.1. get_node_type 関数

ノードの種類を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_node_type(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, node_type)

status : 実行結果を表すステータスコード

node_type : ノードの種類

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeType](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, node_type = cam_device.genapi.get_node_type('Width')
print('status={0} , node_type={1}'.format(status, node_type))
```

7.7.1.2. get_access_mode 関数

ノードのアクセスモードを取得します。

[構文]

```
pytelicam.GenApiWrapper.get_access_mode(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, access_mode)

status : 実行結果を表すステータスコード

access_mode : アクセスモード

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeAccessMode](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, access_mode = cam_device.genapi.get_access_mode('Width')
print('status={0} , access_mode={1}'.format(status, access_mode))
```

7.7.1.3. get_visibility 関数

ノードの可視性（機能にアクセスできるユーザーのレベル）を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_visibility(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名（文字列）

[戻り値]

tuple 型のデータ（status, visibility）

status : 実行結果を表すステータスコード

visibility : 可視性（機能にアクセスできるユーザーのレベル）

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeVisibility](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, visibility = cam_device.genapi.get_visibility('Width')
print('status={0} , visibility={1}'.format(status, visibility))
```

7.7.1.4. get_caching_mode 関数

ノードのキャッシュ設定を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_caching_mode(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, caching_mode)

status : 実行結果を表すステータスコード

caching_mode : キャッシュ設定

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeCachingMode](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, caching_mode = cam_device.genapi.get_caching_mode('Width')
print('status={0} , caching_mode={1}'.format(status, caching_mode))
```

7.7.1.5. get_description 関数

ノードの説明を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_description(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, description)

status : 実行結果を表すステータスコード

description : ノードの説明

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, description = cam_device.genapi.get_description('Width')
print('status={0} , description={1}'.format(status, description))
```

7.7.1.6. get_tool_tip 関数

ノードのツールチップ（簡単な説明）を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_tool_tip(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名（文字列）

[戻り値]

tuple 型のデータ (status, tool_tip)

status : 実行結果を表すステータスコード

tool_tip : ノードの説明

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, tool_tip = cam_device.genapi.get_tool_tip('Width')
print('status={0} , tool_tip={1}'.format(status, tool_tip))
```

7.7.1.7. get_representation 関数

ノードの推奨表現を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_representation(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, representation)

status : 実行結果を表すステータスコード

representation : 推奨表現

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeRepresentation](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, representation = cam_device.genapi.get_representation('Width')
print('status={0} , representation={1}'.format(status, representation))
```

7.7.1.8. get_unit 関数

ノードの単位名を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_unit(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, unit)

status : 実行結果を表すステータスコード

unit : 単位名

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

使用例を以下に示します。

```
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('status={0} , unit={1}'.format(status, unit))
```

7.7.2. ICategory 型関数

7.7.2.1. get_num_of_features 関数

ICategory 型ノードが持つ子 Feature ノードの数を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_num_of_features(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ICategory 型のノード名 (文字列)

[戻り値]

tuple 型のデータ (status, num)

status : 実行結果を表すステータスコード

num : Feature ノード数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、ICategory 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, feature_num = cam_device.genapi.get_num_of_features('Root')
print('status={0} , num={1}'.format(status, feature_num))
```

7.7.2.2. get_feature_name 関数

ICategory 型ノードが持つ子 Feature ノードの名前を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_feature_name(self, node_name, index)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ICategory 型のノード名 (文字列)
index (int)	ノードのインデックス (0 以上の整数値)

[戻り値]

tuple 型のデータ (status, feature_name)

status : 実行結果を表すステータスコード

feature_name : Feature ノード名

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、ICategory 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, node_name = cam_device.genapi.get_feature_name('Root', 0)
print('status={0} , node_name ={1}'.format(status, node_name))
```

7.7.2.3. get_available_feature_names 関数

有効な子 Feature ノードの名前のタプル（リスト）を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_available_feature_names(  
    self,  
    node_name,  
    enable_beginner=True,  
    enable_expert=True,  
    enable_guru=True,  
    enable_invisible=True)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ICategory 型のノード名（文字列）
enable_beginner (bool)	Visibility（可視性）が Beginner に設定されたノードの有効／無効設定。 True に設定された場合、Beginner ノードをタプルに追加します。
enable_expert (bool)	Visibility（可視性）が Expert に設定されたノードの有効／無効設定。 True に設定された場合、Expert ノードをタプルに追加します。
enable_guru (bool)	Visibility（可視性）が Guru に設定されたノードの有効／無効設定。 True に設定された場合、Guru ノードをタプルに追加します。
enable_invisible (bool)	Visibility（可視性）が Invisible に設定されたノードの有効／無効設定。 True に設定された場合、Invisible ノードをタプルに追加します。

[戻り値]

Feature ノードの名前のタプル型データ

[戻り値の型]

tuple(str, ...)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

[備考]

本関数は、ICategory 型ノード専用です。それ以外のノードで実行すると、戻り値のタプルには何も登録されません。

利用不可の Feature ノードの名前は戻り値のタプルに登録されません。

使用例を以下に示します。

```
feature_names = cam_device.genapi.get_available_feature_names('Root')  
print(feature_names)
```

7.7.3. 共通関数

7.7.3.1. get_feature_value 関数

Integer 型、IFloat 型、Boolean 型、IEnumeration 型、IString 型ノードの現在の値を文字列として取得します。

[構文]

```
pytelicam.GenApiWrapper.get_feature_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型、IFloat 型、Boolean 型、IEnumeration 型、IString 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, trigger_mode = cam_device.genapi.get_feature_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))

status, gain_value = cam_device.genapi.get_feature_value('Gain')
print('status={0} , gain_value={1}'.format(status, gain_value))

status, user_defined_name = cam_device.genapi.get_feature_value('UserDefinedName')
print('status={0} , user_defined_name={1}'.format(status, user_defined_name))
```


7.7.3.2. set_feature_value 関数

Integer 型、IFloat 型、Boolean 型、IEnumeration 型、IString 型ノードの値を文字列で設定します。

[構文]

```
pytelicam.GenApiWrapper.set_feature_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)
value (str)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功。
CamApiStatus.GenICamError : GenICam GenApi からエラーがリターンされました。

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型、IFloat 型、Boolean 型、IEnumeration 型、IString 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenICamError](#) が発生したときは、[CameraDevice](#) オブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_feature_value('TriggerMode', 'Off')
print('status={0}'.format(status))

status = cam_device.genapi.set_feature_value('Gain', '1.0')
print('status={0}'.format(status))

status = cam_device.genapi.set_feature_value('UserDefinedName', 'Test')
print('status={0}'.format(status))
```

7.7.4. Integer 型関数

7.7.4.1. get_int_min 関数

Integer 型ノードの現在有効な最小値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_int_min(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, min_value)

status : 実行結果を表すステータスコード

min_value : 最小値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, width_min = cam_device.genapi.get_int_min('Width')
print('gstatus={0} , width_min={1}'.format(status, width_min))
```

7.7.4.2. get_int_max 関数

Integer 型ノードの現在有効な最大値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_int_max(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, max_value)

status : 実行結果を表すステータスコード

max_value : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, width_max = cam_device.genapi.get_int_max('Width')
print('status={0} , width_max={1}'.format(status, width_max))
```

7.7.4.3. get_int_inc 関数

Integer 型ノードの現在有効な増加量を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_int_inc(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, inc_value)

status : 実行結果を表すステータスコード

inc_value : 増加量

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, width_inc = cam_device.genapi.get_int_inc('Width')
print('status={0} , width_inc={1}'.format(status, width_inc))
```

7.7.4.4. get_int_value 関数

Integer 型ノードの値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_int_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, width = cam_device.genapi.get_int_value('Width')
print('status={0} , width={1}'.format(status, width))
```

7.7.4.5. set_int_value 関数

Integer 型ノードの値を設定します。

[構文]

```
pytelicam.GenApiWrapper.set_int_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)
value (int)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenICamError : GenICam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、Integer 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenICamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_int_value('Width', 320)
print('status={0}'.format(status))
if status == pytelicam.CamApiStatus.GenICamError :
    error_message = cam_device.get_last_genicam_error()
    print('error_message ={0}'.format(error_message))
```

7.7.5. IFloat 型関数

7.7.5.1. get_float_min 関数

IFloat 型ノードの現在有効な最小値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_min(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, min_value)

status : 実行結果を表すステータスコード

min_value : 最小値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), float)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, exposure_min = cam_device.genapi.get_float_min('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_min={0}[{1}]'.format(exposure_min, unit))
```

7.7.5.2. get_float_max 関数

IFloat 型ノードの現在有効な最大値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_max(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, max_value)

status : 実行結果を表すステータスコード

max_value : 最大値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), float)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, exposure_max = cam_device.genapi.get_float_max('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_max={0}[{1}]'.format(exposure_max, unit))
```

7.7.5.3. get_float_has_inc 関数

IFloat 型ノードの増加量を取得・設定する機能が実装されているかどうかを取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_has_inc(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, has_inc)

status : 実行結果を表すステータスコード

has_inc : 機能の実装(True)/未実装(False)を表す bool 値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, has_inc = cam_device.genapi.get_float_has_inc('ExposureTime')
print('has_inc={0}'.format(has_inc))
```

7.7.5.4. get_float_inc 関数

IFloat 型ノードの増加量を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_inc(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, inc_value)

status : 実行結果を表すステータスコード

inc_value : 増加量

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), float)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

7.7.5.5. get_float_display_notation 関数

IFloat 型ノードの文字表示方法を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_display_notation(self, display_notation)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, display_notation)

status : 実行結果を表すステータスコード

display_notation : 文字表示方法

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeDisplayNotation](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, display_notation = \
    cam_device.genapi.get_float_display_notation('ExposureTime')
print(' display_notation={0}'.format(display_notation))
```

7.7.5.6. get_float_display_precision 関数

IFloat 型ノードの値を文字に変換する時に使用する小数点以下の精度を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_display_precision(self, display_precision)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, display_precision)

status : 実行結果を表すステータスコード

display_precision : 精度

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, display_precision = \  
    cam_device.genapi.get_float_display_precision('ExposureTime')  
print(' display_precision={0}'.format(display_precision))
```

7.7.5.7. get_float_value 関数

IFloat 型ノードの値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_float_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), float)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, exposure_time = cam_device.genapi.get_float_value('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_time={0}[{1}]'.format(exposure_time, unit))
```

7.7.5.8. set_float_value 関数

IFloat 型ノードの値を設定します。

[構文]

```
pytelicam.GenApiWrapper.set_float_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)
value (float)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenICamError : GenICam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IFloat 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenICamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
# Set ExposureTime to 100[ms]
status = cam_device.genapi.set_float_value('ExposureTime', 100000.0)
print('status={0}'.format(status))
if status == pytelicam.CamApiStatus.GenICamError :
    error_message = cam_device.get_last_genicam_error()
    print('error_message ={0}'.format(error_message))
```

7.7.6. IBoolean 型関数

7.7.6.1. get_bool_value 関数

IBoolean 型ノードの値を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_bool_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : 現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), bool)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IBoolean 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, lut_enable = cam_device.genapi.get_bool_value('LUTEnable')
print('status={0} , lut_enable={1}'.format(status, lut_enable))
```

7.7.6.2. set_bool_value 関数

IBoolean 型ノードの値を設定します。

[構文]

```
pytelicam.GenApiWrapper.set_bool_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)
value (bool)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenICamError : GenICam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IBoolean 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenICamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_bool_value('LUTEnable', True)
print('status={0}'.format(status))
```

7.7.7. IEnumeration 型

7.7.7.1. get_enum_int_value 関数

IEnumeration 型ノードの値を整数値で取得します。

[構文]

```
pytelicam.GenApiWrapper.get_enum_int_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ノードの現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, trigger_mode = cam_device.genapi.get_enum_int_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))
```

7.7.7.2. set_enum_int_value 関数

IEnumeration 型ノードの値を整数値で設定します。

[構文]

```
pytelicam.GenApiWrapper.set_enum_int_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)
value (int)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenlCamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_enum_int_value('TriggerMode', 1)
print('status={0}'.format(status))
```

7.7.7.3. get_enum_str_value 関数

IEnumeration 型ノードの値を文字列で取得します。

[構文]

```
pytelicam.GenApiWrapper.get_enum_str_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : ノードの現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, trigger_mode = cam_device.genapi.get_enum_str_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))
```

7.7.7.4. set_enum_str_value 関数

IEnumeration 型ノードの値を文字列で設定します。

[構文]

```
pytelicam.GenApiWrapper.set_enum_str_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)
value (str)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenlCamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
print('status={0}'.format(status))
```

7.7.7.5. get_available_enum_entry_names 関数

IEnumeration 型ノードが持つ有効な Entry ノードの名前のタプル（リスト）を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_available_enum_entry_names(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名（文字列）

[戻り値]

Entry ノードの名前のタプル型データ

[戻り値の型]

tuple(str, ...)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行すると、戻り値のタプルには何も登録されません。

利用不可の Entry ノードの名前は戻り値のタプルに登録されません。

使用例を以下に示します。

```
entry_names = cam_device.genapi.get_available_enum_entry_names('TriggerMode')
print(entry_names)
```

7.7.7.6. get_num_of_enum_entries 関数

IEnumeration 型ノードのエントリー数を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_num_of_enum_entries(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)

[戻り値]

tuple 型のデータ (status, num)

status : 実行結果を表すステータスコード

entry_num : エントリー数

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
```

7.7.7.7. get_enum_entry_access_mode 関数

IEnumeration 型ノードが保有する IEnumEntry 型ノードリスト内のインデックスを指定し、IEnumEntry 型ノードのアクセスモードを取得します。

[構文]

```
pytelicam.GenApiWrapper.get_enum_entry_access_mode(self, node_name, index)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)
index (int)	指定された IEnumeration 型ノードで管理されている IEnumEntry 型ノードリスト内のインデックス

[戻り値]

tuple 型のデータ (status, access_mode)

status : 実行結果を表すステータスコード

access_mode : アクセスモード

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.NodeAccessMode](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, access_mode = cam_device.genapi.get_enum_entry_access_mode( \
        'TriggerMode', index)
    print('status={0} , access_mode={1}'.format(status, access_mode))
```

7.7.7.8. get_enum_entry_int_value 関数

IEnumeration 型ノードが保有する IEnumEntry 型ノードリスト内のインデックスを指定し、IEnumEntry 型ノードの値を整数値で取得します。

[構文]

```
pytelicam.GenApiWrapper.get_enum_entry_int_value(self, node_name, index)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)
index (int)	指定された IEnumeration 型ノードで管理されている IEnumEntry 型ノードリスト内のインデックス

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : IEnumEntry 型ノードの値 (整数値)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), int)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, value = cam_device.genapi.get_enum_entry_int_value( \
        'TriggerMode', index)
    print('status={0} , value ={1}'.format(status, value))
```


7.7.7.9. get_enum_entry_str_value 関数

IEnumeration 型ノードが保有する IEnumEntry 型ノードリスト内のインデックスを指定し、IEnumEntry 型ノードの値を文字列で取得します。

[構文]

```
pytelicam.GenApiWrapper.get_enum_entry_int_value(self, node_name, index)
```

[パラメータ]

パラメータ	内 容
node_name (str)	IEnumeration 型のノード名 (文字列)
index (int)	指定された IEnumeration 型ノードで管理されている IEnumEntry 型ノードリスト内のインデックス

[戻り値]

tuple 型のデータ (status, str)

status : 実行結果を表すステータスコード

value : IEnumEntry 型ノードの値 (文字列)

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IEnumeration 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, value = cam_device.genapi.get_enum_entry_str_value( \
        'TriggerMode', index)
    print('status={0} , value ={1}'.format(status, value))
```

7.7.8. ICommand 型関数

7.7.8.1. execute_command 関数

ICommand 型ノードのコマンドを実行します。

[構文]

```
pytelicam.GenApiWrapper.execute_command(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、ICommand 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenlCamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_enum_str_value('UserSetSelector', 'UserSet1')
print('status={0}'.format(status))

status = cam_device.genapi.execute_command('UserSetSave')
print('status={0}'.format(status))

while True:
    status, is_done = cam_device.genapi.get_command_is_done('UserSetSave')
    print('status={0}, is_done={1}'.format(status, is_done))
    if status == pytelicam.CamApiStatus.Success and is_done == True:
        break;
    time.sleep(0.2)
```

7.7.8.2. `get_command_is_done` 関数

ICommand 型ノードのコマンド実行状態を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_command_is_done(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名 (文字列)

[戻り値]

tuple 型のデータ (status, value)

status : 実行結果を表すステータスコード

value : コマンドの実行状態。True のとき実行完了、False のとき実行中です。

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、ICommand 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenICamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

7.7.9. IString 型関数

7.7.9.1. get_str_value 関数

IString 型ノードの値（文字列）を取得します。

[構文]

```
pytelicam.GenApiWrapper.get_str_value(self, node_name)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名（文字列）

[戻り値]

tuple 型のデータ（status, value）

status : 実行結果を表すステータスコード

value : 現在の値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), str)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IString 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

使用例を以下に示します。

```
status, user_defined_name =  
cam_device.genapi.get_str_value('UserDefinedName')  
print('status={0} , user_defined_name={1}'.format(status, user_defined_name))
```

7.7.9.2. set_str_value 関数

IString 型ノードの値（文字列）を設定します。

[構文]

```
pytelicam.GenApiWrapper.set_str_value(self, node_name, value)
```

[パラメータ]

パラメータ	内 容
node_name (str)	ノード名（文字列）
value (str)	設定する値

[戻り値]

実行結果を表すステータスコード。

主な戻り値は以下のとおりです。その他の戻り値については [pytelicam.CamApiStatus](#) をご覧ください。

CamApiStatus.Success : 書き込み成功
CamApiStatus.GenlCamError : GenlCam GenApi からエラーがリターンされました

[戻り値の型]

[pytelicam.CamApiStatus](#)

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。

通常のエラー発生時は、戻り値にステータスコードが格納され、例外は送出されません。

[備考]

本関数は、IString 型ノード専用です。それ以外のノードで実行するとエラーステータスがリターンされます。

[CamApiStatus.GenlCamError](#) が発生したときは、[CameraDevice](#) クラスのオブジェクトの [get_last_genicam_error\(\)](#) 関数でエラー情報を取得することができます。

使用例を以下に示します。

```
status = cam_device.genapi.set_str_value('UserDefinedName', 'Test')
print('status={0}'.format(status))
```

7.7.10. その他関数

7.7.10.1. get_access_module 関数

GenApiWrapper クラスのメソッドでアクセスするモジュール（xml ファイル）の設定を取得します。
GenTL インターフェースでオープンしているカメラに対してのみ有効です。

[構文]

```
pytelicam.GenApiWrapper.get_access_module(self)
```

[戻り値]

tuple 型のデータ（status, access_module）
status : 実行結果を表すステータスコード
access_module : 現在のモジュール設定値

[戻り値の型]

tuple([pytelicam.CamApiStatus](#), [pytelicam.AccessModuleType](#))

[例外]

[pytelicam.PytelicamError](#) : この API で予期せぬエラーが発生したとき送出されます。
通常のエラー発生時は、戻り値の status にステータスコードが格納され、例外は送出されません。

7.7.10.2. set_access_module 関数

GenApiWrapper クラスのメソッドでアクセスするモジュール（xml ファイル）を設定します。
GenTL インターフェースでオープンしているカメラに対してのみ有効です。

[構文]

```
pytelicam.GenApiWrapper.set_access_module(self, access_module)
```

[パラメータ]

パラメータ	内 容
access_module (str)	設定するアクセスモジュールの値

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

ストリームインターフェースがオープンされていない場合は、[pytelicam.AccessModuleType.Stream](#) が設定されていても GenApiWrapper クラスの関数を使用してストリームモジュールの情報は取得できません

7.8. SignalHandle クラス

シグナルを管理するクラスです。

TeliCamSDK のシグナル関数にアクセスするためのハンドルを管理します。

このクラスは、以下のシグナルを検知するために使用されます。

- カメラからの画像データ取得完了
- カメラからのイベントデータ取得完了
- カメラの取り外し検知

カメラからの画像データ受信およびカメライベント通知（メッセージ）受信などのシグナルを検知するために使用します。

このクラスのオブジェクトは、[create_signal\(\)](#) 関数で作成されます。

【構文】

pytelicam.SignalHandle

【プロパティ】

	名 称	内 容
	handle (PyCapsule)	シグナルハンドルの値


7.9. CamSystemInfo クラス

pytelicam ライブラリのシステム情報を提供するクラスです。

【構文】

pytelicam.CamSystemInfo

【プロパティ】

	名 称	内 容
	dll_version (str)	ロードされた TeliCamAPI のバージョン













7.10. CameraInfo クラス

カメラ情報を提供するクラスです。

[構文]

pytelicam.CameraInfo

[プロパティ]

	名 称	内 容
	cam_type (pytelicam.CameraType)	カメラのインターフェースタイプ
	cam_vendor (str)	カメラのベンダー名
	cam_model (str)	カメラのモデル名
	cam_serial_number (str)	カメラのシリアル番号
	cam_version (str)	カメラのバージョン
	cam_user_defined_name (str)	カメラのユーザー定義情報
	cam_display_name (str)	カメラの表示名
	tl_vendor (str)	トランスポートレイヤ API のベンダー名
	tl_model (str)	トランスポートレイヤ API のモデル名
	tl_version (str)	トランスポートレイヤ API のバージョン
	tl_display_name (str)	トランスポートレイヤ API の表示名
	tl_if_display_name (str)	トランスポートレイヤ API のインターフェースの表示名




7.11. ImageData クラス

画像データを管理するクラスです。















[構文]

```
pytelicam.ImageData
```

[関数]

	名 称	説 明
	release	ロックを開放し、使用が終了したことを API に通知します。
	get_ndarray	取得した画像データを NumPy 配列で取得します。
	get_memoryview	取得した画像データを Python の memoryview オブジェクトで取得します。

[プロパティ]

	名 称	内 容
	timestamp (int)	カメラが出力する画像データのタイムスタンプ。 USB3 カメラ のとき、単位は nsec です。 GigE カメラにおいては、タイムスタンプの単位はモデルによって異なります。"GevTimestampTickFrequency" ノードの値（単位：Hz）を読み出して逆数を取る事で、タイムスタンプの単位を知ることができます。
	pixel_format (pytelicam.CameraPixelFormat)	カメラが出力する画像データのピクセルフォーマット
	size_x (int)	カメラが出力する画像データの水平有効画素数
	size_y (int)	カメラが出力する画像データの垂直有効画素数
	offset_x (int)	カメラが出力する画像データの水平方向開始位置
	offset_y (int)	カメラが出力する画像データの垂直方向開始位置
	padding_x (int)	カメラが出力する水平画素 パディング（単位：byte）
	block_id (int)	カメラが出力する画像データのフレーム番号。 カメラは画像を出力するたびにフレーム番号をインクリメントします。 画像停止でクリアされます。
	buffer_ptr (PyCapsule)	画像データバッファへのポインタ。 画像データバッファは本 API で管理されています。
	size (int)	ストリーミングバッファに格納された画像データのサイズ（単位：byte）
	image_id (int)	本 API が管理している画像番号。 画像番号は、画像の正常受信・異常受信にかかわらずインクリメントされます。 通常はカメラが設定する block_id を使用します。
	status (pytelicam.CamApiStatus)	画像取得時のステータス
	lock_buffer_index (int)	ロックしている API のストリーミングバッファのバッファインデックス
	camera_device (pytelicam.CameraDevice)	本 ImageData を作成したカメラデバイスのオブジェクト

7.11.1. release 関数

[get_next_image\(\)](#) , [get_current_buffered_image\(\)](#), [get_buffered_image\(\)](#) 関数で取得した [ImageData](#) オブジェクトを開放し、使用が終了したことを API に通知します。

[構文]

```
pytelicam.ImageData.release(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

すべての [ImageData](#) オブジェクトは本 API 内部のストリームリングバッファで管理されています。開放せずに画像データの取り込みを繰り返した場合、ストリームリングバッファに未使用の [ImageData](#) オブジェクトがなくなり、新しい画像データを取得することができなくなります。 使用後は本関数で速やかに開放してください。

[ImageData](#) オブジェクトはスマートポインタで管理されています。 python の `del` 構文、または `with` 構文でオブジェクトを破棄しても構いません。

7.11.2. get_ndarray 関数

取得した画像データを NumPy 配列で取得します。

[構文]

```
pytelicam.ImageData.get_ndarray(  
    self,  
    output_type=pytelicam.OutputImageType.RAW,  
    bayer_conv_mode=pytelicam.BayerConversionMode.BiLinear)
```

[パラメータ]

パラメータ	内 容
output_type (pytelicam.OutputImageType)	出力フォーマット
bayer_conv_mode (pytelicam.BayerConversionMode)	バイヤー変換処理方法。 output_type にRAWが指定されている時、およびカメラからの取得した画像データがバイヤーフォーマット以外 の時はこのパラメータは意味を持ちません。

[戻り値]

画像データ

[戻り値の型]

numpy.ndarray

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

バイヤー変換処理方法の ACPI 法は、BiLinear 法に比べて処理時間がかかります。

バイヤー変換処理は、RAW データを取得した後に OpenCV 等の画像処理ソフトウェアで実行することも可能です。

7.11.3. get_memoryview 関数

取得した画像データを python の memoryview オブジェクトで取得します。

[構文]

```
pytelicam.ImageData.get_memoryview()
```

[戻り値]

画像データ

[戻り値の型]

memoryview

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。




7.12. EventData クラス

カメライベント通知（メッセージ）のデータを管理するクラスです。





[構文]

```
pytelicam.EventData
```

[関数]

	名 称	説 明
	release	ロックを開放し、使用が終了したことを API に通知します。
	get_ndarray	カメライベント通知（メッセージ）データを NumPy 配列で取得します。
	get_memoryview	カメライベント通知（メッセージ）データを Python の memoryview オブジェクトで取得します。

[プロパティ]

	名 称	内 容
	request_id (int)	カメラが出力するカメライベント通知（メッセージ）のリクエスト ID
	event_id (int)	カメラが出力するカメライベント通知（メッセージ）のイベント ID。 イベント ID は、カメラの取扱説明書で確認することができます。
	timestamp (int)	カメラが出力する画像データのタイムスタンプ。 USB3 カメラのとき、単位は nsec です。 GigE カメラにおいては、タイムスタンプの単位はモデルによって異なります。"GevTimestampTickFrequency" ノードの値（単位：Hz）を読み出して逆数を取る事で、タイムスタンプの単位を知ることができます。
	status (pytelicam.CamApiStatus)	カメライベント通知（メッセージ）取得時のステータス

7.12.1. release 関数

[get_event_data\(\)](#) 関数で取得した [EventData](#) オブジェクトを開放し、使用が終了したことを API に通知します。

[構文]

```
pytelicam.EventData.release(self)
```

[戻り値]

なし

[例外]

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

[備考]

イベントデータの使用が終了したら、使用済の [EventData](#) オブジェクトはすみやかに開放してください。 [EventData](#) オブジェクトはスマートポインタで管理されています。 python の `del` 構文、または `with` 構文でオブジェクトを破棄しても構いません。

すべての [EventData](#) オブジェクトは本 API 内部のイベント FIFO バッファで管理されています。 開放せずにカメライベント通知（メッセージ）の取り込みを繰り返した場合、イベント FIFO バッファに未使用の [EventData](#) オブジェクトがなくなり、新しいカメライベント通知（メッセージ）を取得することができなくなります。

7.12.2. get_ndarray 関数

取得したカメライベント通知（メッセージ）データを NumPy 配列で取得します。

【構文】

`pytelicam.EventData.get_ndarray()`

【戻り値】

カメライベント通知（メッセージ）データ

【戻り値の型】

`numpy.ndarray`

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。

7.12.3. get_memoryview 関数

取得したカメライベント通知（メッセージ）データを python の memoryview オブジェクトで取得します。

【構文】

`pytelicam.EventData.get_memoryview()`

【戻り値】

カメライベント通知（メッセージ）データ

【戻り値の型】

`memoryview`

【例外】

[pytelicam.PytelicamError](#) : この API でエラーが発生したとき送出されます。



7.13. PytelicamError クラス

pytelicam ライブラリで例外が発生したときに送出されるクラスです。

【構文】

pytelicam.PytelicamError

【プロパティ】

	名 称	内 容
	message (str)	例外のメッセージ
	status (pytelicam.CamApiStatus)	例外が送出された原因のエラーステータスコード

8. 列挙型の説明

公開される列挙型は以下のとおりです。

[列挙型]

	名 称	内 容
	CamApiStatus	ステータスコードを表す列挙型
	CameraType	カメラのインターフェースタイプを表す列挙型
	CameraAccessMode	カメラのアクセスモード（特権）を表す列挙型
	CameraPixelFormat	カメラの画像データのピクセルフォーマットを表す列挙型
	CameraAcquisitionMode	画像データ転送モードを表す列挙型
	CameraEventType	カメライベント通知（メッセージ）の種類を表す列挙型
	NodeType	GenICam ノードの種類を表す列挙型。
	NodeAccessMode	GenICam ノードのアクセスモードを表す列挙型
	AccessModuleType	GenApiWrapper クラスでアクセスするモジュール（xml ファイル）を表す列挙型
	OutputImageType	画像データの出力フォーマットを表す列挙型
	BayerConversionMode	画像データのバイヤー変換処理方式を表す列挙型
	CameraImageFormat	カメラの映像フォーマットを表す列挙型
	CameraPixelFormat	カメラの映像ストリームのピクセルフォーマットを表す列挙型
	CameraTestPattern	カメラのテストパターンを表す列挙型
	CameraAcqFrameRateCtrl	カメラの映像のフレームレート設定を表す列挙型
	CameraImageBufferMode	カメラのイメージバッファモード設定を表す列挙型
	CameraTriggerSequence	カメラの露光時間制御モードを表す列挙型
	CameraTriggerSource	カメラのランダムトリガシャッタのトリガソースを表す列挙型
	CameraTriggerActivation	カメラのハードウェアトリガの有効エッジを表す列挙型
	CameraExposureTimeCtrl	カメラの露光時間 制御モードを表す列挙型
	CameraLineSelector	カメラのラインを表す列挙型
	CameraLineSource	カメラのラインの信号種類を表す列挙型
	CameraLineMode	カメラのライン入出力を表す列挙型
	CameraTimerTriggerSource	カメラの Timer0Active 信号の基準信号を表す列挙型
	CameraGainAuto	カメラの AGC 動作モードを表す列挙型
	CameraBalanceRatioSelector	カメラのホワイトバランスゲイン設定の対象となる要素を表す列挙型
	CameraBalanceWhiteAuto	カメラのホワイトバランスゲイン自動調整モードを表す列挙型
	CameraSaturationSelector	カメラの彩度の要素を表す列挙型
	CameraSaturationSelector	カメラの彩度の要素を表す列挙型
	CameraColorCorrectionMatrixSelector	カメラの色補正マトリックス調整機能の要素を表す列挙型

	名 称	内 容
	CameraUserSetSelector	カメラのユーザー設定機能におけるユーザー設定チャンネルの要素を表す列挙型
	CameraChunkSelector	カメラのチャンクの要素を表す列挙型
	CameraFrameSynchronization	カメラのフレーム同期制御方法のタイプを表す列挙型

【備考】

本マニュアルに記載されていない列挙型は、未サポートの列挙型です。

8.1. CamApiStatus 列挙型（ステータスコード）

ステータスコードを表す列挙型です。

[構文]

```
pytelicam.CamApiStatus
```

[列挙子リスト]

列挙子	内 容
Success	成功しました。
NotInitialize	API の初期化が行われていません。
AlreadyInitialized	API の初期化がすでに行われています。
NotFound	カメラが見つかりません。
AlreadyOpened	すでにオープンされています。
AlreadyActivated	すでにアクティブ化されています。
InvalidCameraIndex	指定されたカメラインデックスが不正です。
InvalidCameraHandle	指定されたカメラハンドルが不正です。
InvalidNodeHandle	指定されたノードハンドルが不正です。
InvalidStreamHandle	指定されたストリームハンドルが不正です。
InvalidRequestHandle	指定されたリクエストハンドルが不正です。
InvalidEventHandle	指定されたイベントハンドルが不正です。
InvalidParameter	指定されたパラメータが不正です。
BufferTooSmall	指定されたバッファが小さすぎます。
NoMemory	本 API 内部のメモリ確保に失敗しました。
MemoryNoAccess	指定されたメモリへのアクセスに失敗しました。
NotImplemented	機能がカメラまたはAPIに実装されていません。 レジスタ名（ノード名）が間違っている場合にもこのエラーが発生します。
Timeout	タイムアウトが発生しました。
CameraNotResponding	カメラを検出できませんでした。カメラを接続しているケーブルが外れている可能性があります。接続を確認してください。
EmptyCompleteQueue	完了したリクエストが、完了キューに存在しません。
NotReady	準備が完了していません。
AccessModeSetError	アクセスモードの設定に失敗しました。
InsufficientBufferNum	バッファの数が不足しています。
IoDeviceError	コントローラから I/O エラーが通知されました。
LogicalParameterError	論理エラーが通知されました。
NotConnectedToUSB3	USB3 で接続されていません。
XML_LoadError	XML ファイルのロードに失敗しました。
GenICamError	GenICam エラーが発生しました。
DLL_LoadError	DLL ファイルのロードに失敗しました。
NoSystemResources	システム リソースが不足しているため、要求されたサービスを完了できませんでした。
InvalidAddress	無効なアドレスが指定されました。
WriteProtect	指定アドレスがライトプロテクトされているため、処理に失敗しました。
BadAlignment	不整列なアドレスにアクセスしました。
AccessDenied	アクセスが拒否されました。
Busy	ビジー状態です。
NotReadable	読み出し可能な状態ではありません。
NotWritable	書き込み（実行）可能な状態ではありません。
NotAvailable	設定パラメータが有効ではありません。
VerifyError	カメラのレジスタにデータを書き込んだとき、ベリファイエラーが発生しました。
RequestTimeout	リクエストがタイムアウトしました。

ResendTimeout	再送要求を行わない場合、あるパケットを受信してから時間内に次のパケットを受信できず、タイムアウトしました。（GigE カメラのみ）
ResponseTimeout	再送要求を行ってから時間内に要求したパケットを受信できず、タイムアウトしました。（GigE カメラのみ）
BufferFull	受信データのサイズが指定の最大値を超えました。
UnexpectedBufferSize	実際の受信データサイズが、トレーラに記載されているサイズと一致しませんでした。
UnexpectedNumber	受信パケットが最大数を超えました。
PacketStatusError	パケットのステータスがエラーでした。
ResendNotImplemented	カメラがパケット再送コマンドに対応していません。
PacketUnavailable	ストリームパケットは利用できません。
MissingPackets	フレームデータの完了前に、次のブロックのリーダーが受信されました。（パケットが欠落している可能性があります。）
FlushRequested	ユーザー操作により、リクエストがフラッシュされました。
TooManyPacketMissing	パケットの消失が規定値を超えました。 このエラーが GigE カメラで発生するとき、ネットワークの帯域が不足している可能性があります。ジャンボフレームを有効にしたり、カメラのフレームレートを低下させて、帯域幅を超えないようにしてください。
FlushedByD0Exit	パワーステートが変更されて、リクエストがフラッシュされました。
FlushedByCameraRemove	カメラが取り外されて、リクエストがフラッシュされました。
Driver_LoadError	ドライバのロードに失敗しました。
MappingError	ユーザーバッファをシステム空間の仮想アドレスにマッピングする際にエラーが発生しました。 システムリソースが少ないために発生した可能性があります。
FileOpenError	ファイルのオープンに失敗しました。
FileWriteError	ファイルへのデータ書き込みに失敗しました。
FileReadError	ファイルからデータ読み出しに失敗しました。
FileNotFound	ファイルが見つかりませんでした。
InvalidParameterFromCam	カメラから無効なパラメータエラーがリターンされました。
PayloadSizeNotAligned	ストリームインターフェースのペイロードサイズ設定にアライメントされていない異常がありました。
DataDiscarded	データブロックのいくつかのデータが破棄されました。（USB3 カメラのみ）
DataOverrun	カメラはすべてのデータを送信することができませんでした。 SIRM レジスタの設定に問題がある可能性があります。
OutOfMemory	システムまたはシステム内の他のハードウェア (フレームグラバー) でメモリが不足しました。
U3vNotAvailable	USB3 DLL のロードに失敗しました。（警告） USB3 以外のインターフェースは利用可能です。
GevNotAvailable	GigE DLL のロードに失敗しました。（警告） GigE 以外のインターフェースは利用可能です。
GenTLNotAvailable	GenTL DLL のロードに失敗しました。（警告） GenTL 以外のインターフェースは利用可能です。
Unsuccessful	その他のエラーが発生しました。

8.2. CameraType 列挙型

カメラのインターフェースタイプを表す列挙型です。

[構文]

pytelicam.CameraType

[列挙子リスト]

列挙子	内 容
All	東芝テリー製デバイスドライバを使用したすべてのインターフェース。GenTL インターフェースは含まれません。
Gev	GigE カメラ インターフェース。
U3v	USB3 カメラ インターフェース。
GenTL	GenTL インターフェース。 フレームグラババー メーカーが提供する GenTL Producer を使用します。
Unknown	不明なインターフェース。

8.3. CameraAccessMode 列挙型

カメラのアクセスモード（特権）を表す列挙型です。

GigE カメラ以外では意味を持ちません。

[構文]

pytelicam.CameraAccessMode

[列挙子リスト]

列挙子	内 容
Open	オープンアクセスモード。 カメラレジスタの読み込みはできますが、書き込みはできません。
Control	コントロールアクセスモード。 カメラの全機能を制御できます。 他のアプリケーションは、オープンアクセスモードでのみカメラのオープンをすることができます。
Exclusive	特別アクセスモード。 カメラの全機能を独占的に制御できます。 他のアプリケーションは、カメラのオープンができなくなります。

8.4. CameraPixelFormat 列挙型

カメラの画像データのピクセルフォーマットを表す列挙型です。

[構文]

```
pytelicam.CameraPixelFormat
```

[列挙子リスト]

列挙子	内 容
Unknown	不明なフォーマット
Mono8	Mono8 フォーマット
Mono10	Mono10 フォーマット
Mono10p	Mono10 packed フォーマット
Mono12	Mono12 フォーマット
Mono12p	Mono12 packed フォーマット
Mono16	Mono16 フォーマット
BayerGR8	BayerGR8 フォーマット
BayerGR10	BayerGR10 フォーマット
BayerGR12	BayerGR12 フォーマット
BayerRG8	BayerRG8 フォーマット
BayerRG10	BayerRG10 フォーマット
BayerRG12	BayerRG12 フォーマット
BayerGB8	BayerGB8 フォーマット
BayerGB10	BayerGB10 フォーマット
BayerGB12	BayerGB12 フォーマット
BayerBG8	BayerBG8 フォーマット
BayerBG10	BayerBG10 フォーマット
BayerBG12	BayerBG12 フォーマット
RGB8	RGB8 フォーマット
BGR8	BGR8 フォーマット
BGR10	BGR10 フォーマット
BGR12	BGR12 フォーマット
YUV411_8	YUV411_8 フォーマット
YUV422_8	YUV422_8 フォーマット
YUV8	YUV8 フォーマット

8.5. CameraAcquisitionMode 列挙型

画像データ転送モードを表す列挙型です。

[構文]

```
pytelicam.CameraAcquisitionMode
```

[列挙子リスト]

<u>CameraAcquisitionMode</u>	内 容
Continuous (連続画像データ 転送モード)	stop() 関数が実行されるまで継続的に画像を撮像・転送するモードです。 通常は、このモードが使用されます。 本関数を実行するとAcquisitionStartコマンドが実行され、画像が取得されます。 カメラのAcquisitionModeレジスタに "Continuous" が設定されます。
SingleFrame (シングルフレーム画像デ ータ転送モード)	1 枚の画像を撮像・転送するモードです。 本関数を実行すると AcquisitionStart コマンドが実行され、画像が取得されます。 新しい画像を取得するときは、本関数を実行してください。 本関数を実行すると カメラのAcquisitionFrameCount レジスタの値は 1に書き換わります。
MultiFrame (マルチフレーム画像デー タ転送モード)	カメラの AcquisitionFrameCount レジスタに設定された枚数の画像を撮像・転送するモードです。 AcquisitionFrameCount レジスタの値を変更する場合は GenApiWrapper クラスのオブジェクトを使用します。 例：cam_device.genapi.set_int_value('AcquisitionFrameCount', 5) 本関数を実行するとAcquisitionStartコマンドが実行され、画像が取得されます。 新しい画像を取得するときは、本関数を実行してください。
ImageBufferRead (カメライメージバッファ 転送モード)	stop() 関数が実行されるまで継続的に画像を撮像し、カメラ内のバッファに保存するモードです。 カメラのImageBufferReadレジスタに実行が指示されるとカメラ内のバッファから画像が転送されます。 カメラの ImageBufferRead レジスタへ実行を指示するには GenApiWrapper クラスのオブジェクトを使用します。 例：cam_device.genapi.execute_command('ImageBufferRead')

8.6. CameraEventType 列挙型

カメライベント通知（メッセージ）の種類を表す列挙型です。

[構文]

pytelicam.CameraEventType

[列挙子リスト]

列挙子	内 容
FrameTrigger	トリガ受付
FrameTriggerError	トリガエラー
FrameTriggerWait	トリガ受付待ち開始
FrameTransferStart	画像データ転送開始
FrameTransferEnd	画像データ転送終了
ExposureStart	露光開始
ExposureEnd	露光終了
Timer0Start	Timer0 開始
Timer0End	Timer0 終了
ALCLatestInformation	ALC 動作更新時の値
ALCConverged	ALC 動作収束時の値
Unknown	不明なカメライベント

8.7. NodeType 列挙型

GenICam ノードの種類を表す列挙型です。

[構文]

pytelicam.NodeType

[列挙子リスト]

列挙子	内 容
Value	Value ノード(IValue)
Base	Base ノード(IBase)
Integer	Integer ノード(IInteger)
Boolean	Boolean ノード(ICollection)
Command	Command ノード(ICommand)
Float	Float ノード(IFloat)
String	String ノード(IString)
Register	Register ノード(IRegister)
Category	Category ノード(ICategory)
Enumeration	Enumeration ノード(IEnumeration)
EnumEntry	EnumEntry ノード(IEnumEntry)
Port	Port ノード(IPort)
Unknown	不明なノード

8.8. NodeAccessMode 列挙型

GenICam ノードのアクセスモードを表す列挙型です。

[構文]

pytelicam.NodeAccessMode

[列挙子リスト]

列挙子	内 容
NotImplemented	未実装
NotAvailable	利用不可
WriteOnly	書き込み専用
ReadOnly	読み出し専用
ReadAndWrite	読み書き可能
Undefined	オブジェクト未初期化
Unknown	不明なアクセスモード

8.9. NodeVisibility 列挙型

ノードの可視性（機能にアクセスできるユーザーの推奨レベル）を表す列挙型です。

[構文]

pytelicam.NodeVisibility

[列挙子リスト]

列挙子	内 容
Beginner	常に可視
Expert	熟練者および、専門家に可視
Guru	専門家に可視
Invisible	不可視
Undefined	オブジェクト未初期化
Unknown	不明な可視性

8.10. NodeCachingMode 列挙型

ノードのキャッシュ設定を表す列挙型です。

[構文]

pytelicam.NodeCachingMode

[列挙子リスト]

列挙子	内 容
NoCache	キャッシュ不可
WriteThrough	レジスタ書き込み時にキャッシュへも書き込み
WriteAround	読み出し時にキャッシュへ書き込み。 書き込み時はレジスタのみへ書き込み。
Undefined	オブジェクト未初期化
Unknown	不明なキャッシュ設定

8.11. NodeRepresentation 列挙型

ノードの推奨表現を表す列挙型です。

[構文]

pytelicam.NodeRepresentation

[列挙子リスト]

列挙子	内 容
Linear	リニア表示のスライダ
Logarithmic	対数表現のスライダ
Boolean	チェックボックス
PureNumber	10 進数編集コントロール
HexNumber	16 進数編集コントロール
IPV4Address	IP アドレス形式表示
MACAddress	MAC アドレス形式表示
Undefined	オブジェクト未初期化
Unknown	不明

8.12. NodeDisplayNotation 列挙型

ノードの文字表示方法を表す列挙型です。

[構文]

pytelicam.NodeDisplayNotation

[列挙子リスト]

列挙子	内 容
Automatic	固定小数点と指数表記で短い表記を自動選択
Fixed	固定小数点
Scientific	指数表記
Undefined	オブジェクト未初期化
Unknown	不明

8.13. AccessModuleType 列挙型

GenApiWrapper クラスでアクセスするモジュール（xml ファイル）を表す列挙型です。

[構文]

pytelicam.AccessModuleType

[列挙子リスト]

列挙子	内 容
RemoteDevice	リモートデバイスモジュール（カメラ）
System	GenTL Producer のシステムモジュール
Interface	GenTL Producer のインターフェースモジュール
Device	GenTL Producer のデバイスモジュール
Stream	GenTL Producer のストリームモジュール

8.14. OutputImageType 列挙型

画像データの出力フォーマットを表す列挙型です。

【構文】

pytelicam.OutputImageType

【列挙子リスト】

列挙子	内 容
Raw	Raw フォーマット
Bgr24	BGR 24bit フォーマット
Bgra32	BGRA 32bit フォーマット

8.15. BayerConversionMode 列挙型

画像データのバイヤー変換処理方式を表す列挙型です。

【構文】

pytelicam.BayerConversionMode

【列挙子リスト】

列挙子	内 容
BiLinear	BiLinear 法
Acpi	ACPI 法

8.16. CameraImageFormat 列挙型

カメラの映像フォーマットを表す列挙型です。

【構文】

pytelicam.CameraImageFormat

【列挙子リスト】

列挙子	内 容
Format0	Format0
Format1	Format1
Format2	Format2

8.17. CameraTestPattern 列挙型

カメラのテストパターンを表す列挙型です。

[構文]

pytelicam.CameraTestPattern

[列挙子リスト]

列挙子	内 容
Off	テストパターン Off、通常映像
Black	全てのピクセルが 0
White	全てのピクセルが 255 @Mono8
GreyA	全てのピクセルが 170 @Mono8
GreyB	全てのピクセルが 85 @Mono8
HorizontalRamp	水平方向ランブ
GreyScale	グレースケール
ColorBar	カラーバー
VerticalRamp	垂直方向ランブ

8.18. CameraAcqFrameRateCtrl 列挙型

カメラの映像のフレームレート設定を表す列挙型です。

[構文]

pytelicam.CameraAcqFrameRateCtrl

[列挙子リスト]

列挙子	内 容
NoSpecify	ExposureTime の設定値優先
Manual	AcquisitionFrameRate の設定値優先

8.19. CameraImageBufferMode 列挙型

カメラのイメージバッファモード設定を表す列挙型です。

[構文]

pytelicam.CameraImageBufferMode

[列挙子リスト]

列挙子	内 容
Off	ExposureTime の設定値優先
On	AcquisitionFrameRate の設定値優先

8.20. CameraTriggerSequence 列挙型

カメラの露光時間制御モードを表す列挙型です。

[構文]

pytelicam.CameraTriggerSequence

[列挙子リスト]

列挙子	内 容
Sequence0	TriggerSequence0 (Edge モード) 露光時間は電子シャッタの設定値 I IDC2 規格に準拠したカメラ : TriggerSequence レジスタ = TriggerSequence0 I IDC2 規格に準拠していないカメラ : ExposureMode レジスタ = Timed TriggerSelector レジスタ = FrameStart
Sequence1	TriggerSequence1 (Level モード) 露光時間はトリガ信号のパルス幅 I IDC2 規格に準拠したカメラ : TriggerSequence レジスタ = TriggerSequence1 I IDC2 規格に準拠していないカメラ ￥ : ExposureMode レジスタ = TriggerWidth TriggerSelector レジスタ = FrameStart
Sequence6	TriggerSequence6 (Bulk / FrameBurst モード) 1 回のトリガ信号入力で、連続して複数回の露光と映像出力を行います。 I IDC2 規格に準拠したカメラ : TriggerSequence レジスタ = TriggerSequence6 I IDC2 規格に準拠していないカメラ : ExposureMode レジスタ = Timed TriggerSelector レジスタ = FrameBurstStart

8.21. CameraTriggerSource 列挙型

カメラのイメージバッファモード設定を表す列挙型です。

[構文]

pytelicam.CameraTriggerSource

[列挙子リスト]

列挙子	内 容
Line0	Hardware Trigger Line0
Line1	Hardware Trigger Line1
Line2	Hardware Trigger Line2
Software	ソフトウェアトリガ

8.22. CameraTriggerActivation 列挙型

カメラのハードウェアトリガの有効エッジを表す列挙型です。

[構文]

pytelicam.CameraTriggerActivation

[列挙子リスト]

列挙子	内 容
FallingEdge	Falling Edge モード
RisingEdge	Rising Edge モード

8.23. CameraExposureTimeCtrl 列挙型

カメラの露光時間 制御モードを表す列挙型です。

[構文]

pytelicam.CameraExposureTimeCtrl

[列挙子リスト]

列挙子	内 容
NoSpecify	AcquisitionFrameRate の設定値優先 IIDC2 規格に準拠したカメラ : ExposureTimeControl レジスタ = NoSpecify IIDC2 規格に準拠していないカメラ : 利用不可
Manual	ExposureTime の設定値優先 IIDC2 規格に準拠したカメラ : ExposureTimeControl レジスタ = Manual IIDC2 規格に準拠していないカメラ : ExposureAuto レジスタ = Off
Auto	自動露光時間制御 IIDC2 規格に準拠したカメラ : ExposureTimeControl レジスタ = Auto IIDC2 規格に準拠していないカメラ : ExposureAuto レジスタ = Continuous

8.24. CameraLineSelector 列挙型

カメラのラインを表す列挙型です。

[構文]

pytelicam.CameraLineSelector

[列挙子リスト]

列挙子	内 容
Line0	Line0
Line1	Line1
Line2	Line2

8.25. CameraLineSource 列挙型

カメラのラインの信号種類を表す列挙型です。

[構文]

pytelicam.CameraLineSource

[列挙子リスト]

列挙子	内 容
Off	汎用出力は無効です。
VD	VD 同期信号です。
UserOutput	UserOutputValueAll にて設定した値を出力します。
Timer0Active	ストロボ制御用信号として使用できます。トリガ入力からの遅延量と幅を設定できます。
AcquisitionActive	AcquisitionStart 状態であることを示す信号です。
FrameTriggerWait	ランダムトリガシャッタ時に、トリガ待ち受け期間であることを示す信号です。
FrameActive	露光開始から CMOS 転送完了までの期間です。
FrameTransferActive	映像をバスに転送している期間です。
ExposureActive	露光開始から露光終了までの期間です。

8.26. CameraLineMode 列挙型

カメラのライン入出力を表す列挙型です。

[構文]

pytelicam.CameraLineMode

[列挙子リスト]

列挙子	内 容
Output	出力
Input	入力

8.27. CameraTimerTriggerSource 列挙型

カメラの Timer0Active 信号の基準信号を表す列挙型です。

[構文]

pytelicam.CameraTimerTriggerSource

[列挙子リスト]

列挙子	内 容
Off	Timer 出力は無効です。
Line0Active	TRIG_IN(Line0)入力より Timer がスタートします。
FrameTrigger	トリガ受付より Timer がスタートします。
ExposureStart	露光開始より Timer がスタートします。
ExposureEnd	露光終了より Timer がスタートします。

8.28. CameraGainAuto 列挙型

カメラの ADC 動作モードを表す列挙型です。

【構文】

pytelicam.CameraGainAuto

【列挙子リスト】

列挙子	内 容
Off	マニュアルゲイン制御(MANUAL)
Auto	自動ゲイン制御(AGC)

8.29. CameraBalanceRatioSelector 列挙型

カメラのホワイトバランスゲイン設定の対象となる要素を表す列挙型です。

【構文】

pytelicam.CameraBalanceRatioSelector

【列挙子リスト】

列挙子	内 容
Blue	BalanceRatio = Blue Gain
Red	BalanceRatio = Red Gain

8.30. CameraBalanceWhiteAuto 列挙型

カメラのホワイトバランスゲイン自動調整モードを表す列挙型です。

【構文】

pytelicam.CameraBalanceWhiteAuto

【列挙子リスト】

列挙子	内 容
Off	待機状態
Continuous	ホワイトバランスゲインを自動調整しつづけます。
Once	一度だけホワイトバランスゲインを自動調整します。

8.31. CameraSaturationSelector 列挙型

カメラの彩度の要素を表す列挙型です。

[構文]

pytelicam.CameraSaturationSelector

[列挙子リスト]

列挙子	内 容
U	Saturation = U Gain
V	Saturation = V Gain

8.32. CameraColorCorrectionMatrixSelector 列挙型

カメラの色補正マトリックス調整機能の要素を表す列挙型です。

[構文]

pytelicam.CameraColorCorrectionMatrixSelector

[列挙子リスト]

列挙子	内 容
RG	SelectorI = R , SelectorJ = G
RB	SelectorI = R , SelectorJ = B
GR	SelectorI = G , SelectorJ = R
GB	SelectorI = G , SelectorJ = B
BR	SelectorI = B , SelectorJ = R
BG	SelectorI = B , SelectorJ = G

8.33. CameraUserSetSelector 列挙型

カメラのユーザー設定機能におけるユーザー設定チャンネルの要素を表す列挙型です。

[構文]

```
pytelicam.CameraUserSetSelector
```

[列挙子リスト]

列挙子	内 容
Default	工場出荷時設定（ロードのみ）
UserSet1	ユーザー設定チャンネル 1
UserSet2	ユーザー設定チャンネル 2
UserSet3	ユーザー設定チャンネル 3
UserSet4	ユーザー設定チャンネル 4
UserSet5	ユーザー設定チャンネル 5
UserSet6	ユーザー設定チャンネル 6
UserSet7	ユーザー設定チャンネル 7
UserSet8	ユーザー設定チャンネル 8
UserSet9	ユーザー設定チャンネル 9
UserSet10	ユーザー設定チャンネル 10
UserSet11	ユーザー設定チャンネル 11
UserSet12	ユーザー設定チャンネル 12
UserSet13	ユーザー設定チャンネル 13
UserSet14	ユーザー設定チャンネル 14
UserSet15	ユーザー設定チャンネル 15

8.34. CameraChunkSelector 列挙型

カメラのチャンクの要素を表す列挙型です。

[構文]

```
pytelicam.CameraChunkSetSelector
```

[列挙子リスト]

列挙子	内 容
BlockID	BlockID
FrameBurstTriggerCount	FrameBurstTriggerCount
SequentialShutterNumber	SequentialShutter number
SequentialShutterElement	SequentialShutter element
UserArea	User area
ExposureTime	Exposure time
Gain	Gain
WhiteBalanceR	WhiteBalanceR
WhiteBalanceB	WhiteBalanceB
LineStatusAll	LineStatusAll

8.35. CameraFrameSynchronization 列挙型

カメラのフレーム動機制御方法のタイプを表す列挙型です。

[構文]

pytelicam.CameraFrameSynchronization

[列挙子リスト]

列挙子	内 容
Bus	バス同期
Off	内部同期

9. サンプルソース

サンプルコードを理解するためには GenICam、GigE Vision、USB3 Vision、IIDC2 レジスタマップなどの知識が必要になる場合があります。これらの規格に関する詳細情報、最新情報はそれぞれの規格提供元のホームページを参照してください。

GenICam <https://www.emva.org/standards-technology/genicam/>
GigE Vision <https://www.automate.org/a3-content/vision-standards-gige-vision>
USB3 Vision <https://www.automate.org/a3-content/usb3-vision-standard>
IIDC2 http://jiaa.org/mv_dl/iidc2/

サンプルコードに関する詳細情報は、"pytelicam Samples Manual Jpn.pdf"を参照してください。

10. その他

10.1. 免責事項

pytelicam ライブラリの免責事項は、TeliCamSDK の免責事項に準じます。

TeliCamSDK の免責事項は、TeliCamSDK に付属の“License Agreement TeliCamSDK Jpn.pdf”に記載されています。必ずご一読の上、ご利用されますようお願い致します。

[TeliCamSDK インストールフォルダ]\Licenses フォルダを参照ください。

10.2. ライセンス

pytelicam ライブラリのライセンスは、TeliCamSDK のライセンスに準じます。

TeliCamSDK は、複数の独立したソフトウェアコンポーネントを使用しています。個々のソフトウェアコンポーネントは、それぞれ第三者の著作権が存在します。

TeliCamSDK は、第三者が規定したエンドユーザーライセンスアグリーメントあるいは著作権通知（以下、「EULA」といいます）に基づきフリーウェアとして配布されるソフトウェアコンポーネントを使用しております。

「EULA」の中には、実行形式のソフトウェアコンポーネントを配布する条件として、当該コンポーネントのソースコードの入手を可能とするよう求めているものがあります。当該「EULA」の対象となるソフトウェアコンポーネントのお問い合わせに関しては、9.4 項に記載の方法でお問い合わせください。

TeliCamSDK で使用している、対象となるソフトウェアコンポーネントの「EULA」は以下のディレクトリにインストールされています。

[TeliCamSDK インストールフォルダ]\Licenses

東芝テリー株式会社は、東芝テリー株式会社が定める条件の基で TeliCamSDK の動作を保証します。（“License Agreement TeliCamSDK Jpn.pdf” と “License Agreement TeliCamSDK Sample Jpn.pdf” をご覧ください。）ただし、「EULA」に基づいて配布されるソフトウェアコンポーネントには、著作権者または弊社を含む第三者の保証がないことを前提に、お客様がご自身でご利用になられることが認められるものであります。この場合、当該ソフトウェアコンポーネントは無償でお客様に使用許諾されますので、適用法令の範囲内で、当該ソフトウェアコンポーネントの保証は一切ありません。ここでいう保証とは、市場性や特定目的適合性についての黙示の保証も含まれますが、それに限定されるものではありません。当該ソフトウェアコンポーネントの品質や性能に関するすべてのリスクはお客様が負うものとします。また、当該ソフトウェアコンポーネントに欠陥があると分かった場合、それに伴う一切の派生費用や修理・訂正に要する費用は、東芝テリー株式会社は一切の責任を負いません。適用法令の定め、または書面による合意がある場合を除き、著作権者や上記許諾を受けて当該ソフトウェアコンポーネントを使用したこと、または使用できないことに起因する一切の損害について何らの責任も負いません。著作権者や第三者が、そのような損害の発生する可能性について知らされていた場合でも同様です。なお、ここでいう損害には、通常損害、特別損害、偶発損害、間接損害が含まれます（データの消失、またはその正確さの喪失、お客様や第三者が被った損失、他のソフトウェアとのインターフェースの不適合化等も含まれますが、これに限定されるものではありません）。当該ソフトウェアコンポーネントの使用条件や遵守いただかなければならない事項等の詳細は、各「EULA」をお読みください。

TeliCamSDK で使用している「EULA」の対象となるソフトウェアコンポーネントは、以下の表のとおりです。これらのソフトウェアコンポーネントをお客様自身でご利用いただく場合は、対応する「EULA」をよく読んでから、ご利用くださるようお願いいたします。

対応ソフトウェアモジュール	ライセンス
GenICam GenApi	GenICam License

GenICam GenApi は以下のサードパーティソフトウェアを使用しています。

対応ソフトウェアモジュール	ライセンス
MathParser	LGPLv2.1
Log4Cpp	LGPLv2.1
CppUnit	LGPLv2.1
CLSerAll	NI license
xs3p	DSTC license
xxhash	xxhash license
XSLTProc	MIT license
XSDe	Proprietary

pytelicam ライブラリは以下のサードパーティソフトウェアを使用しています。

対応ソフトウェアモジュール	ライセンス
pybind11	BSD-style license
numpy	BSD license
wheel	MIT license

TeliCamSDK は、LGPL 適用ソフトウェアのバイナリを再配布しており、これらのソースコードに限っては、LGPL の定めに従い、入手、改変、再配布する権利をお客様は有します。ソースコードはご希望のお客さまへは、メディア（CD-ROM 等）に書き込み郵送にてお送りします。送料等実費にてご提供させていただいておりますので、ご希望の場合は 9.4 項に記載の方法でお問い合わせください。尚、ソースコードは、お客様が入手の権利を有するオープンソースソフトウェアのみ配布いたします。（TeliCamSDK のソースコードは含まれません。）ソースコードの内容などについてのご質問にはお答えいたしかねますので、あらかじめご了承ください。

Microsoft、Windows、Windows XP、Windows Vista、Windows 7、Windows 8.1、Windows 10、Windows 11 及び Visual C++ は、Microsoft 社の商標もしくは登録商標です。
GigE Vision は、AIA (Automated Imaging Association)の各国における商標または登録商標です。
USB3 Vision は、AIA (Automated Imaging Association)の各国における商標または登録商標です。
GenICam は、EMVA (European Machine Vision association)の各国における商標または登録商標です。
CoaXPress は、JIIA (Japan Industrial Imaging Association)の登録商標です。
その他、本ドキュメントに記載の会社名、団体名、製品名、規格名等の名称は、各社各団体における商標または登録商標です。

10.3. 改定履歴

Date	Version	内容
2021/04/13	1.0.0	初版
2021/11/09	1.0.1	<ul style="list-style-type: none">関数追加 save_parameter(), load_parameter()CamApiStatus 列挙型の列挙子を追加CameraPixelFormat 列挙型の列挙子を追加
2022/10/27	1.0.2	<ul style="list-style-type: none">関数追加 create_device_object_from_ip_address(), get_next_image_with_trigger()その他、誤記の修正等
2023/07/24	1.1.0	<ul style="list-style-type: none">CameraControl クラス追加CameraImageFormat 列挙子を追加CameraTestPattern 列挙子を追加CameraAcqFrameRateCtrl 列挙子を追加CameraImageBufferMode 列挙子を追加CameraTriggerSequence 列挙子を追加CameraTriggerSource 列挙子を追加CameraTriggerActivation 列挙子を追加CameraExposureTimeCtrl 列挙子を追加CameraLineSelector 列挙子を追加CameraLineSource 列挙子を追加CameraLineMode 列挙子を追加CameraTimerTriggerSource 列挙子を追加CameraGainAuto 列挙子を追加CameraBalanceRatioSelector 列挙子を追加CameraBalanceWhiteAuto 列挙子を追加CameraSaturationSelector 列挙子を追加CameraColorCorrectionMatrixSelector 列挙子を追加CameraUserSetSelector 列挙子を追加CameraChunkSelector 列挙子を追加CameraFrameSynchronization 列挙子を追加その他、誤記の修正等
2024/12/13	1.1.1	<ul style="list-style-type: none">9. サンプルソースの説明追加その他、誤記の修正等

10.4. お問い合わせ

TeliCamSDK ならびにGigE カメラ、USB3 カメラ、CoaXPress カメラに関するよくあるご質問とその回答(FAQ)は、[弊社ホームページ](#)の「サポート」－「産業カメラに関するFAQ」サイトをご利用ください。

それでも解決できない場合は、[弊社ホームページ](#)の「お問い合わせ」サイトに記載の電話番号またはメールフォームにてご連絡ください。